VŠB – Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Aplikace pro rozvoj kognitivních a motorických dovedností pomocí tabletu

Application for Cognitive and Motor Skills Training

2015 Jakub Simandl

VŠB - Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Zadání diplomové práce

Student: **Bc. Jakub Simandl**

Studijní program: N2647 Informační a komunikační technologie

Studijní obor: 2612T025 Informatika a výpočetní technika

Téma: Aplikace pro rozvoj kognitivních a motorických dovedností pomocí tabletu
Application for Cognitive and Motor Skills Training

Zásady pro vypracování:

Mobilní zařízení dnes nalézají uplatnění i v rámci terapie motorických a mentálních handicapů. Cílem této práce je vyvinout aplikaci pomáhající rozvoji kognitivních a motorických dovedností formou nácviku a opakování jednoduchých cviků a úkolů.

1. Rešerše počítačových aplikací užívaných pro rozvoj motorických a mentálních handicapů.
2. Výběr vhodných metod a forem tréninku kognitivních (např. n-back) a motorických funkcí (např. obkreslování figur).
3. Implementace zvolených metod na zařízení s OS Android.
4. Tvorba UI a vykreslování grafiky pomocí OpenGL ES.
5. Testování aplikace na různých typech zařízení (verze Androidu, rozlišení).

Seznam doporučené odborné literatury:

[1] Reto Meier, Professional Android 4 Application Development, Wrox, 2012, ISBN-13: 978-1118102275
[2] Cay S. Horstmann, Core Java(TM), Volume I-Fundamentals, Prentice Hall, 2007, ISBN-13: 978-0132354769
[3] Solanto, M. V., Cognitive-Behavioral Therapy for Adult ADHD: Targeting Executive Dysfunction, The Guilford Press, 2011, ISBN 1609181314

Formální náležitosti a rozsah diplomové práce stanoví pokyny pro vypracování zveřejněné na webových stránkách fakulty.

Vedoucí diplomové práce: **Mgr. Ing. Michal Krumnikl, Ph.D.**

Datum zadání: 01.09.2013
Datum odevzdání: 07.05.2015

doc. Dr. Ing. Eduard Sojka
vedoucí katedry

prof. RNDr. Václav Snášel, CSc.
děkan fakulty

Souhlasím se zveřejněním této diplomové práce dle požadavků čl. 26, odst. 9 *Studijního a zkušebního řádu pro studium v magisterských programech VŠB-TU Ostrava.*

V Ostravě 11. března 2015 .....

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

V Ostravě 11. března 2015 .....

Tímto bych chtěl poděkovat zejména vedoucímu své diplomové práce panu Mgr. Ing. Michalovi Krumniklovi, Ph.D., za trpělivost, věcné připomínky a odborné vedení. Dále bych chtěl poděkovat panu Ing. Jiřímu Krajíčkovi a odborným pedagogům komunity iSEN, za ochotu při testování aplikace.

Abstrakt

V dnešní době je rozvoj kognitivních schopností jedince s využitím moderních technologií, jako jsou například dotyková zařízení, velmi oblíbeným jak v terapeutických centrech po celém světě, tak i u jednotlivců toužících zlepšit si své schopnosti. V této práci prozkoumáme současný stav dostupných programů a aplikací pro kognitivní rozvoj a implementujeme vzorovou aplikaci pro dotyková zařízení operačního systému Android s využitím grafického standardu OpenGL ES 2.0. V rámci práce poté zhodnotíme testování použitelnosti aplikace koncovými uživateli, ke kterému došlo ve spolupráci s komunitou iSEN.

Klíčová slova: Android, OpenGL ES, tablet, kognitivní rehabilitace, CACR, použitelnost

Abstract

The development of cognitive abilities of individuals using modern technology such as touch devices is nowadays very popular and widely used by therapeutical centers around the world, but also by individuals who desire to improve their skills. In this thesis, we examine current status of available programs and applications for cognitive developement and then we implement sample application aimed at touch devices running the Android operating system using a graphical standard OpenGL ES 2.0. We evaluate the usability of the created user interface of the application by means of user testing, which took place this year in cooperation with iSEN comunity.

Keywords: Android, OpenGL ES, tablet, cognitive retraining, CACR, usability

Seznam použitých zkratek a symbolů

ABI	–	Acquired Brain Injury
ADD	–	Attention-Deficit Disorder
ADHD	–	Attention Deficit Hyperactivity Disorder
API	–	Application Programming Interface
ART	–	Android Runtime
CACR	–	Computer-Assisted Cognitive-Retraining
CPU	–	Central Processing Unit
DVM	–	Dalvik Virtual Machine
CPU	–	Central Processing Unit
FMRI	–	Functional Magnetic Resonance Imaging
GPU	–	Graphics Processor Unit
JAR	–	Java Archive
JVM	–	Java Virtual Machine
PAS	–	Poruchy Autistického Spektra
PC	–	Personal Computer
SDK	–	Software Developement Kit
VBO	–	Vertex Buffer Object

Seznam tabulek

Seznam obrázků

Seznam výpisů zdrojového kódu

Obsah

1 Úvod

Počítačem asistovaný rozvoj kognitivních schopností je v dnešní době velmi používaný v terapiích, ve vzdělávacích organizacích po celém světě, ale i rekreačně u jedinců, kteří se snaží o rozvoj a zlepšení svých kognitivních schopností.

V této diplomové práci se budu zabývat tvorbou aplikace pro rozvoj kognitivních funkcí a to konkrétně aplikace pro operační systém Android, jeden z nejrozšířenějších operačních systémů pro mobilní zařízení současnosti. Implementace aplikace pro mobilní platformy, zejména tablety a mobilní telefony, mi umožní využít ovládání aplikace dotykovými gesty, které je v mnoha případech pro koncového uživatele schůdnější, než ovládání klasických počítačových aplikací pomocí vstupu myši a klávesnice, případně jiných periferií. V práci se dále budu zabývat tvorbou jednoduchého aplikačního frameworku pro rychlejší vývoj těchto aplikací pomocí grafické knihovny OpenGL ES, která je běžně dostupná právě na zmíněných mobilních zařízeních. Framework by měl v sobě zahrnovat jak práci s grafikou (vykreslování jednoduché 2D grafiky, 3D grafiky, textu, a reprezentace prvků uživatelského rozhraní), která je podstatná pro grafickou reprezentaci námi implementované aplikace, tak zpracování uživatelských dotyků, přehrávání zvuku, jakožto pomocného indikátoru různých akcí uživatele, a podobně.

V první kapitole práce se zaměřím na popsání současného vývoje v oblasti počítačově asistovaného rozvoje kognitivních schopností. Budu se snažit seznámit čtenáře se současnou podobou této oblasti, jejími trendy a také tím, zda jsou představené metody vhodné a používané pro rozvoj kognitivních funkcí jedince.

Ve druhé kapitole provedu rešerši dostupných aplikací výše uvedeného zaměření, jejich roztřídění na jednotlivé oblasti lidských kognitivních schopností, které se aplikace snaží svými metodami zlepšit, případně určím, čím bych se v těchto aplikacích mohl při následné implementaci vzorové aplikace inspirovat. Na trhu se pochopitelně objevuje široká nabídka aplikací, proto půjde jen o jakýsi průřez těch nejpodstatnějších a v dnešní době běžně používaných.

Ve třetí kapitole popíši vybrané metody, které budou následně určeny k implementaci v samotné aplikaci. Metody jsou vybrány buďto na základě jejich prokazatelných účinků na rozvoj kognitivních schopností jedince, případně jejich podobností s takovými metodami a dále také na základě aplikací, jež budou popsány ve druhé kapitole a z nichž v mnohých jsou tyto metody použity. Při výběru metod bude přihlédnuto i na jejich možnost a vhodnost implementace právě pro prostředí tabletů, které umožňuje ovládání dotykovými gesty, jelikož ne všechny typy úkolů jsou pro tento typ uživatelského vstupu uzpůsobeny.

Ve čtvrté kapitole popíši obecně platformu Android, jako platformu pro vývoj aplikací. Stručně nastíním vývoj interaktivních grafických aplikací a specifika platformy, které musí uživatel použít při vývoji pomocí grafického standardu OpenGL ES. Dále popíši vývoj pomocného frameworku zastřešujícího grafickou knihovnu OpenGL ES, která byla použita pro zobrazování samotné aplikace a uživatelského rozhraní, jeho jednotlivé části starající se o různé prvky aplikace, od vykreslování grafiky přes uživatelský vstup až po přehrání zvukových souborů. Cílem frameworku pochopitelně nebude snažit se konkurovat dnešním běžně dostupným frameworkům, které jsou pro vývoj grafických aplikací v prostředí operačního systému Android používané, ale spíše nastínit čtenáři, co tvorba takového frameworku a obecně práce s grafikou na mobilních platformách obnáší, jaké jsou možnosti optimalizace a budoucího vývoje, a na jaké zásady je třeba při práci s knihovnou OpenGL ES brát ohled. V kapitole se nebudu zabývat přímo jednotlivými kroky tvorby grafického výstupu aplikace za použití knihovny OpenGL ES.

V páté kapitole se více zaměřím na uživatelské rozhraní aplikace, jelikož toto rozhraní by mělo splňovat určité kvalitativní požadavky, pakliže bude používáno lidmi s určitým deficitem v některých z kognitivních oblastí, a tedy i možným deficitem v oblastech jemné motoriky a oblasti koordinace ruka-oko, či s možností určitého mentálního deficitu.

V šesté kapitole poté provedu zhodnocení přenositelnosti aplikace na různá zařízení lišící se především velikostí obrazovky, případně i procesorové architektury a provedu zhodnocení toho, zda se při vývoji s těmito různorodými parametry přístrojů mohou vyskytovat nějaké problémy. Popíši uzpůsobení vývoje pro operační systém Android z hlediska přenositelnosti aplikací mezi zařízeními různých rozměrů a nastíním výhody a konstrukty, které nám pro vývoj operační systém Android v této oblasti nabízí.

2 Počítačem asistovaný rozvoj kognitivních schopností

Žijeme v době, kdy jsme obklopeni moderními technologiemi, a proto i do různých oblastí medicíny tyto technologie pronikají. Jednou z těchto oblastí je i oblast rozvoje a obnovy kognitivních schopností jedince, používaná například při poúrazovém zhoršení kognitivních schopností, či při jejich deficitu, způsobeném například různými mentálními poruchami, mírnou kognitivní poruchou a podobně [9]. V rozvoji a obnově kognitivních poruch lze v dnešní době tento technologický trend sledovat. Počítače a pomocné technologie se stávají běžnými v terapiích jedinců, kteří utrpěli poúrazové poškození mozku (ABI - acquired brain injury). Ve Spojených státech více než 73% rehabilitačních center používá tzv. Počítačem Asistovanou Obnovu Kognitivních Schopností (CACR - Computer-Assisted Cognitive-Retraining).

Tyto programy využívají principů kognitivní rehabilitace - procesu zvyšování a vylepšování schopností jedince a jeho kapacity zpracovat a použít příchozí informace. V nich jsou zahrnuty metody pro obnovu kognitivních funkcí a pro trénování kompenzačních technik [8]

Současné počítačové programy se vyvinuly z jednoduchých herních stanic (jako například použití konzole Atari) až po sofistikované nástroje, použité spolu s komplexní strategií a mnohdy v kombinaci s jinými druhy terapie. V dřívějších dobách bylo použití těchto programů v terapiích omezeno pouze na rozvíjení reflexů a vizuálně motorické koordinace. V dnešní době v sobě zahrnují nespočet kognitivních a jazykových cvičení.

Vývoj různorodosti programů byl v podstatě podmíněn tím, že kognitivní poruchy se velmi liší z osoby na osobu, a žádný program by nebyl schopen pokrýt větší množinu těchto poruch. Přes velkou různorodost kognitivních poruch je většina pro jedince velmi invalidizující a vyžaduje specializované a vysoce organizované použití CACR. Zatímco prvotním cílem kognitivní nápravy je zlepšit kognitivní fungování jedince, konečným cílem je ovlivnit jeho denní fungování. Metody tréninku mohou být vybrány a zacíleny tak, aby rozvíjely specifickou kognitivní schopnost s ohledem na osobnostní silné i slabé stránky jedince a unikátní rehabilitační cíle. Základní podstatou kognitivní rehabilitace je tedy i to, že "léčebný plán"musí být ušit na míru specifickým potřebám daného jedince.

Nejúspěšnější programy byly schopny průlomově zlepšit pozornost, schopnost zpracování vizuálních vjemů a také uvažování, případně řešení problémů. Většina programů na zlepšování paměti zahrnuje praktikování opakovaných cvičení, která mají za cíl zvýšit množství informací, které si je člověk schopen v danou chvíli vyvolat. Určitou limitací mezi programy cílenými na zlepšování paměti je, jak se ukázalo, malá přenositelnost dosažených výsledků na běžné funkční schopnosti jedince. I přes to, že jedinec dosáhl dobrých výsledků v počítačovém programu, ještě nutně neznamená, že se to projeví v

lepším fungování i mimo samotnou počítačovou úlohu. Dále je zde mnoho typů deficitů, které nejsou pro počítačový trénink velmi vhodné, jako například schopnost řeči či pragmatika. Tyto většinou vyžadují větší míru interakce mezi lékařem a pacientem [3].

Dvěma základními přístupy k tréninku sloužícímu ke zlepšení kognitivních funkcí je přístup restorativní (obnovovací) a kompenzační. Obnovovací přístup zahrnuje specifické opakování úkonů a cvičení za použití počítačových či webových aplikací, tužky a papíru, či diskuze. Cílem je pomocí opakování a praktických úkonů zlepšit specifickou kognitivní dovednost. Kompenzační strategie učí způsoby, jak daný nedostatek obejít tím, že budeme spoléhat na alternativní metodu. Namísto pokusů o rozvíjení daných omezených kognitivních dovedností jsou učeny různé techniky, které využívají ostatní zdravé smysly nebo externí pomoc a podporu. V obou případech je k danému problému přistoupeno pomocí jakési strategie toho, jak přistoupit k danému úkonu, který využívá některou z kognitivních schopností co nejefektivněji. Existují i hybridní přístupy k nápravě, které kombinují oba zmíněné přístupy.

Existují různé teoretické možnosti přístupu ke struktuře a formátu restorativních nápravných technik. Kupříkladu v tzv. "bottom-up"programech jednotlivá cvičení postupně projdou hierarchií procvičovaných schopností od těch úplně elementárních, jako např. pozornost, čas reakce na podnět (reakční čas) a pracovní paměť, až ke komplexnějším schopnostem, jakými jsou abstraktní uvažování, schopnost sekvencování a řešení problému. V tzv. "top-down"programech jsou komplexní schopnosti (například již zmíněné řešení problému) procvičovány ihned od počátku, s tím, že základní dovednosti jsou procvičovány v rámci těch komplexních. Pro co možná nejlepší výsledky s použitím CACR je potřeba důkladného plánování a pozorování průběžné výkonnosti jedince. Podstatný je také správný výběr softwarových programů. Terapeut musí vědět, k čemu je daný program vhodný a jak zapadá do pacientova současného modelu léčby.

I přesto že se vkládají do počítačem asistovaného rozvoje kognitivních schopností veliké naděje a programy působí slibně, literatura se nezmiňuje o tom, že by CACR bylo obecně lepší nežli tradiční (tedy bez použití počítačových programů) kognitivní rehabilitační techniky. CACR je více efektivní ve zlepšení míry pozornosti při tréninku určité schopnosti než při obecném tréninku nezaměřeném na určitou činnosti nebo typ deficitu [7]. Některé studie dokonce nebyly schopny prokázat významné rozdíly ve zlepšeních schopností mezi dvěma pozorovanými skupinami - jednou používající CACR a druhou, která asistence počítačových programů nevyužívala [1]. Většina výstupu z dnešních výzkumů v této oblasti je založena buďto na testování prováděném na jednotlivcích, případně na statisticky nevýznamných skupinách [2]. Výsledná aplikace diplomové práce by tedy mohla být použita v případných "bottom-up"programech, s tím, že zaměřena bude

zejména na rozvoj pracovní paměti, pozornosti a prostorové představivosti. Tím, že je aplikace určena pro dotyková zařízení, inherentně rozvíjí i jemnou motoriku a schopnost koordinace ruka-oko, jelikož dotykové gesta vyžadují určitou míru preciznosti a snahy ze strany uživatele.

2.1 Výzkum v oblasti tréninku pracovní paměti

Studie se shodují v tom, že většina lidí s poruchami pozornosti má sníženou kapacitu pracovní paměti. To platí i pro poruchy pozornosti v dětství (hyperaktivita s poruchou pozornosti = ADHD - Attention Deficit Hyperactivity Disorder), traumatická poranění mozku nebo pro poruchu pozornosti při přirozeném stárnutí, ale rovněž pro mělčí poruchy koncentrace [10].

Výzkum dále ukazuje, že poruchy pracovní paměti souvisejí rovněž s nižší školní či pracovní úspěšností. A obráceně, dobrá kapacita pracovní paměti se úzce shoduje s vrozenou inteligencí [10].

Neustále se zvyšující počet výzkumných prací dokazuje efektivitu různých tréninků pracovní paměti. Mezi nimi je významná zejména studie [11] potvrzující efektivitu tréninku pracovní paměti u školních dětí s ADHD (hyperaktivita s poruchou pozornosti), a to v multicentrické (prováděné na více pracovištích současně) a placebem kontrolované studii.

Studie prokázala klinicky i statisticky významné efekty tréninku i na ukazatelích kapacity pracovní paměti, které nebyly specificky trénovány. Efekty tréninku se generalizují i na schopnost potlačit impulzivitu a tím dosáhnout lepšího soustředění. Novinkou ve vědeckých výzkumech je prokazatelná souvislost mezi zvýšením kapacity pracovní paměti a zlepšením pozornosti. Pracovní paměť je zásadní jak pro schopnost soustředění, tak pro schopnost přiměřeně odolávat rušivým podnětům z okolí a pro schopnost komplexního uvažování. Zvýšení kapacity pracovní paměti se přenáší i na zlepšení pozornosti a kontrolování impulzivity [11].

2.2 Komunita iSEN

V České republice patří mezi významné uskupení zabývající se využitím počítačových zařízení k rozvoji kognitivních schopností a k podpoře komunikace s osobami trpícími určitým typem kognitivního deficitu, komunita iSEN. Ve svých nabízených kurzech se zaměřuje zejména na vzdělávání dětí se speciálními vzdělávacími potřebami, s mentálním a kombinovaným postižením, či s poruchami autistického spektra, a to využitím právě počítačových tabletů a aplikací k tomuto účelu vytvořených.

V koordinaci s panem Ing. Jiřím Krajíčkem byla navázána spolupráce právě s tímto institutem pro potřeby testování toho, zda vůbec, a do jaké míry bude implementovaná aplikace diplomové práce splňovat určité kvalitativní požadavky uživatelského rozhraní, jež jsou na aplikace podobného zaměření kladeny. Zhodnocením tohoto testování se zabývá pátá kapitola.

3 Dostupné aplikace pro rozvoj kognitivních schopností

V rámci diplomové práce došlo k průzkumu současných aplikací, zaměřujících se na rozvoj kognitivních schopností jedince. V dnešní době se na trhu vyskytuje velké množství aplikací tohoto zaměření. Většina komerčních aplikací je však určena pro konkrétní terapeutické programy, a tak jejich samostatné použití nemusí zaručit téměř žádnou efektivitu v rozvoji daných schopností.

Bylo vybráno několik aplikací, jak komerčních (i aplikace, které jsou součásti komplexnějších programů rozvoje kognitivních schopností), tak volně dostupných, popsáno jejich stručné shrnutí, a také to, na jaké oblasti kognitivních schopností se zaměřují, jaké jsou jejich výhody i nevýhody, případně jaké použité prvky uživatelského rozhraní či samotné metody a úkoly by mohly být inspirací pro naši vzorovou aplikaci. I přesto, že vzorová aplikace bude primárně určena pro ovládání dotykovými gesty, tedy určena pro prostředí tabletu, v rešerši aplikací jsme se zaměřili i na aplikace běžně dostupné pro stolní počítače (desktopové i webové aplikace), ovládané klasickým vstupem v podobě počítačové myši a klávesnice. Je patrné, že dotykové ovládání bude pro koncového uživatele vždy uživatelsky přívětivější a intuitivnější, poskytující naší aplikaci v porovnání s klasickými aplikacemi pro stolní PC značnou výhodu v použitelnosti.

V průzkumu jsou zahrnuty pouze aplikace s uživatelským rozhraním v anglickém a českém jazyce, aplikace v jiných jazycích brány v potaz nebyly. Pro pokrytí většího množství aplikací pro dotyková zařízení byly prozkoumány všechny tři současné nejpoužívanější mobilní platformy - iOS, Android i Windows. Vzorová aplikace diplomové práce však bude cílena pouze na platformu Android.

3.1 Komerčně dostupné aplikace

Komerčně dostupné aplikace jsou většinou součástí komplexních programů rozvoje kognitivních schopností, zahrnující mnohdy před- i po- terapeutické sezení s odborníkem a terapii velmi zacílenou na konkrétního jedince, pro dosažení co možná nejlepších výsledků po tréninku. Až na výjimky neposkytují tyto programy své aplikace pro mobilní zařízení, většinou se jedná o klasické desktopové aplikace

3.1.1 Program Activate

Jedním z podstatných institutů v oblasti rozvoje kognitivních schopností (zejména pozornosti u dětí trpících poruchami pozornosti) ve Spojených státech amerických je firma C8 Sciences. Ve svém produktu Activate nabízí komplexní řešení pro rozvoj kognitivních schopností jedince, čímž se snaží zejména vylepšovat akademický prospěch klientů. V

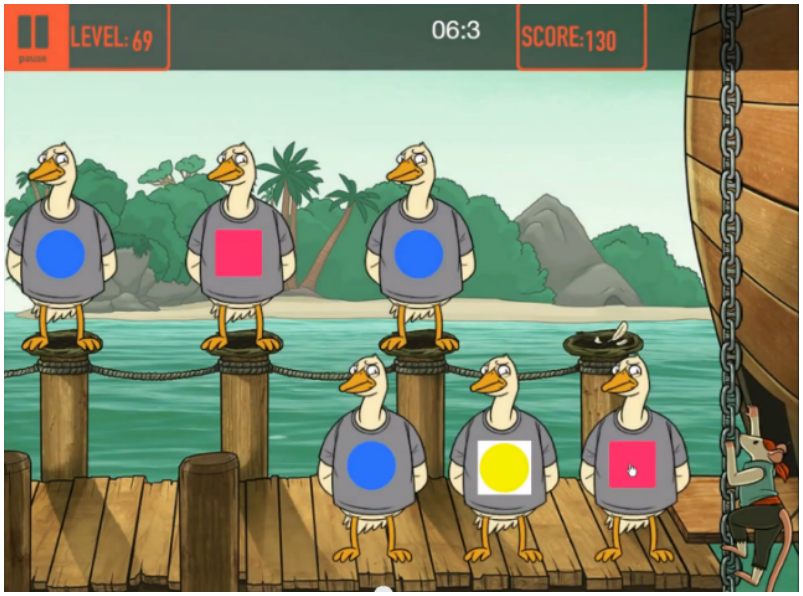
tomto produktu kombinuje fyzické cvičení se speciálním softwarem pro dosažení optimálních výsledků.

Pro tuto diplomovou práci je pochopitelně nepodstatné zkoumat celý komplexní program, jeho softwarová část však může být v určitých směrech inspirací. Softwarová část je určena pro dotykové platformy a to pouze v rámci celého komplexního placeného programu Activate, pro potřeby rešerše tedy není dostupná.

Podstatným prvkem tohoto softwaru je, podle oficiálního webu programu Activate, unikátní schopnost programu (respektive jednotlivých jeho dílčích úkolů) přizpůsobovat se aktuální výkonnosti jedince, a tímto optimalizovat zlepšování jeho kognitivních schopností. V rámci programu byla vyvinuta speciální kritéria, určující to, zda již subjekt dosáhl svého pomyslného vrcholu v dané oblasti a zda je tedy potřeba zvýšit obtížnost. Tímto v podstatě subjekt rychle projde cvičeními, které jsou jeho silnou stránkou a naopak pomaleji těmi, které mu příliš nejdou a zabrání tomu, aby subjekt příliš dlouho pracoval s úrovní cvičení, na které již dosáhl svého maxima.

Software se skládá z několika dílčích úkolů tematicky zaměřených pro cílovou věkovou skupinu (děti prvního a druhého stupně základní školy) do podoby pirátského dobrodružství. Jednotlivé úkoly jsou vždy zaměřeny na určitou kognitivní oblast. Celou aplikací provádí uživatele hlasový průvodce spolu s jasně zřetelnými prvky uživatelského rozhraní, které uživateli poskytují jak pomoc při řešení úkolů, tak zpětnou vazbu při úspěchu či neúspěchu.

Obrázek 1: Activate - včasné nakrmení hladových pirátů se zvednutou paží



Obrázek 2: Activate - doplnění správné posloupnosti

3.1.2 EDA PLAY

Aplikace EDA Play je českou aplikací, která pomáhá uživatelům se zrakovými vadami trénovat zrak a uživatelům s motorickými vadami trénovat jemnou motoriku. Aplikace je navržena tak, aby interaktivní hrou motivovala uživatele ke sledování děje na displeji tabletu a k plnění úkolů. Vizuální i zvukové zpracování aplikace podporuje koordinaci ruky a oka. Aplikace je svým vizuálním zaměřením určena spíše pro děti.

Aplikace nabízí 4 úrovně obtížností úkolů. Ke splnění těch nejjednodušších stačí dotyk na libovolném místě displeje. Druhá úroveň obtížnosti stimuluje koordinaci ruka/oko. Třetí úroveň procvičuje základní obrazce a pohyby rukou v základních směrech. Čtvrtá obtížnost nabízí nejnáročnější úkoly, které procvičují i nepravidelné obrazce, dokreslování nebo spojování na základě podobnosti.

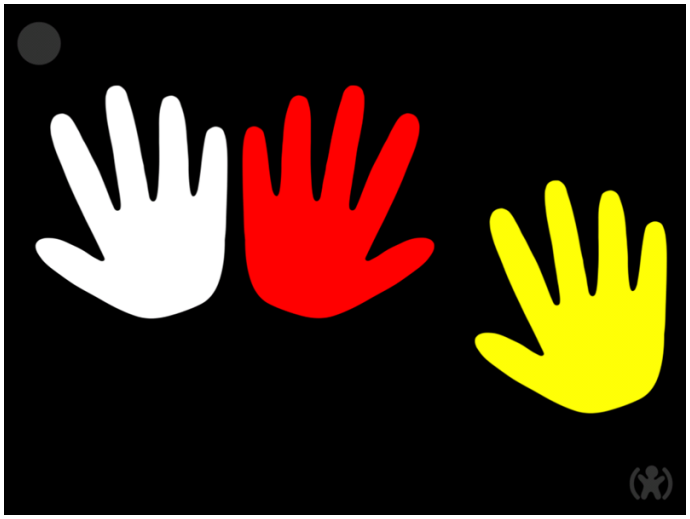
Součástí aplikace je i hlasový průvodce, kdy každý úkol představí milý dětský hlas, který i pochválí za správně provedený úkol a v případě, že se nedaří, povzbuzuje k opakování úkolu. Aplikace je dostupná ve dvou jazykových mutacích - češtině a angličtině.

Dále aplikace obsahuje sekci Dovednosti, která zaznamenává práci uživatele s aplikací. Terapeuti nebo i samotný uživatel tak mohou sledovat vývoj dovedností v čase.

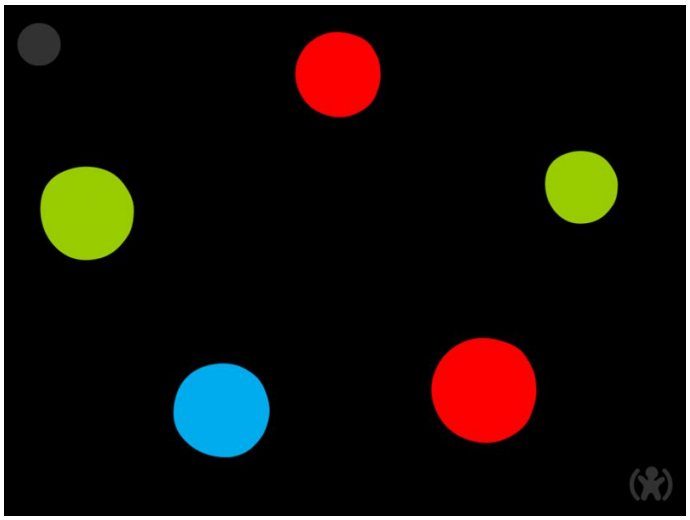
Aplikace je nastavena tak, aby se po jejím spuštění a krátkém úvodu automaticky objevil první úkol. Sekce, které mají ovládat rodiče či terapeuti, jsou skryty pod tlačítky

umístěnými v rozích displeje. Pro vstup do menu je nutné tato tlačítka několik vteřin podržet.

Na první pohled (viz obrázek 3 a obrázek 4) aplikace zaujme velmi jednoduchým a strohým uživatelským rozhraním samotných úkolů. Je tomu pochopitelně kvůli zaměření úkolů na kompenzaci očních vad. Příliš zahlcené uživatelské rozhraní by uživatele (zvláště pokud jsou cílovou skupinou aplikace zejména děti) příliš rozptylovalo a vedlo by k nižší přehlednosti.



Obrázek 3: EDA Play - výběr dané barvy ruky na základě hlasových pokynů



Obrázek 4: EDA Play - spojování kuliček stejné barvy

3.1.3 Cogmed

Program Cogmed je akreditovaný trénink pracovní paměti spolu s počítačovým programem, určený zejména ke zlepšení pozornosti u uživatelů trpících některým z deficitů v oblasti pozornosti (ADHD, ADD). Program je v současné době v české republice nabízen pouze Centrem duševního zdraví, sídlícím v Jeseníku. Program je opravdu komplexním řešením, zahrnujícím úvodní sezení, pětitýdenní trénink s týdenními konzultacemi trenéra, přístupem na tréninkový web Cogmedu, prošetřením po 6 měsících užívání a 12 měsíčním kondičním tréninkem.

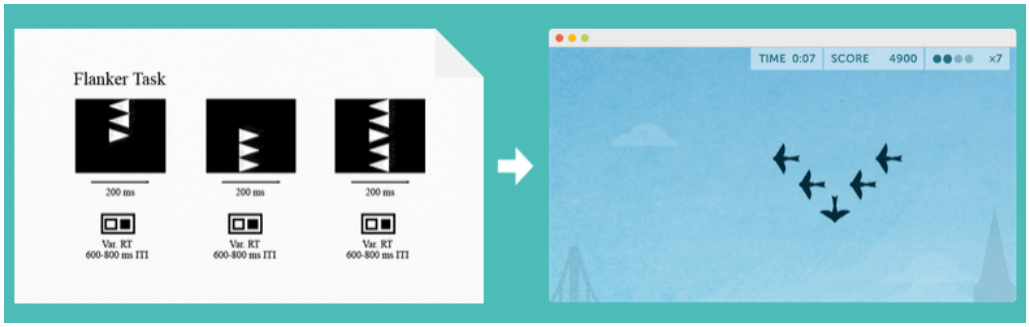
Samotná softwarová část tréninku je určena pro stolní počítače. Opět se jedná o sadu určitých úloh, simulujících dané kognitivní schopnosti jedince, přičemž tyto programy jsou nabízeny ve třech variantách v závislosti na věku jedince a to konkrétně jako Cogmed JM (předškolní věk), Cogmed RM (školní věk) a Cogmed QM (dospělí). Na obrázku 5 lze vidět uživatelské rozhraní aplikace, konkrétně úkol z varianty Cogmed JM založen na psychologickém testu Corsi block-tapping task (4.1.2)

Obrázek 5: Cogmed JM - úkol ve stylu Corsi block-tapping task

3.1.4 Lumosity

Lumosity je rozsáhlou platformou zaměřující se na oblasti tréninku kognitivních schopností. V současnosti tuto platformu využívá podle údajů na samotných webových stránkách aplikace více než 70 milionů uživatelů. Ve své aplikaci pro prostředí webu a operačních systémů Android a iOS nabízí širokou škálu úkolů, zaměřených na jednotlivé oblasti kognitivních schopností, přičemž uživatel si může vytvořit individualizovaný plán roz-

voje. V každé z pěti různých oblastí rozvoje (paměť, pozornost, rychlost, flexibilita, řešení problému) se nachází několik specifičtěji zaměřených oblastí (jako například asociace jmen a tváři u oblasti "paměť", plánování efektivní trasy u oblasti "řešení problému"), ze kterých si uživatel poskládá svůj plán rozvoje. Dle velikosti výběru je poté trénink zpoplatněn. Lumosity je podle tvůrců vytvořen ve spolupráci s neurovědci a upravuje tréninky na základě dat získaných ze své rozsáhlé uživatelské základny. Na obrázku 6 lze vidět použití Eriksen-Flankerova psychologického testu jako základu pro jeden z úkolů v aplikaci.



Obrázek 6: Lumosity - aplikace Eriksen-Flankerova úkolu

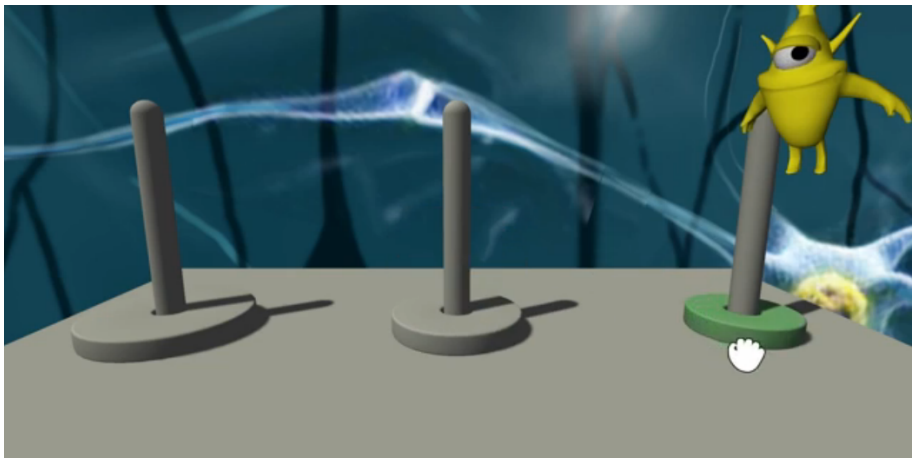
3.2 Volně dostupné aplikace

Trh s volně dostupnými aplikacemi je opravdu obrovský. V rešerši jsme se zaměřili spíše na některé zajímavější projekty, jako například projekt Masarykovy univerzity Newron, jelikož klasických aplikací pro mobilní zařízení je nepřeberné množství. Aplikace se povětšinou nespecializují na konkrétní oblast a snaží se spíše o obecný "trénink mozku"jedince, například o zrychlení jeho reakčních časů, zlepšení krátkodobé paměti a podobně. Na rozdíl od komerčních aplikací běžně používaných v profesionálních terapiích zde pochopitelně zcela chybí individualizovaný přístup pro daného jedince. Poskytují však značnou výhodu v mnohem snadnější dostupnosti, povětšinou disponují jednodušším uživatelským rozhraním, a pochopitelnou výhodou je i jejich cena. V rámci výzkumu jsme až na výjimky nenarazili na volně dostupnou aplikaci určenou pro specializované terapie jedince s určitou formou mentální dysfunkce, aplikace jsou většinou určeny pro zdravé jedince toužící pouze po zlepšení svých schopností, kteří jsou schopni pochopit uživatelské rozhraní a ovládání aplikace.

3.2.1 Newron

Zajímavým projektem, jež je součástí aplikovaného výzkumu Masarykovy univerzity je projekt Newron.

Projekt Newron je terapeutický software se zaměřením na podporu a rozvoj jedinců s různými psychosociálními obtížemi. Jedná se konkrétně o aplikaci pro stolní PC, založenou na několika úkolech, které u uživatele stimulují příslušné kognitivní schopnosti. V současné době se projekt zaměřuje cíleně na děti a dospívající s diagnostikovanou poruchou autistického spektra (PAS), v budoucnu je plánováno rozšíření terapeutického zaměření i na další poruchy (hyperaktivita, poruchy pozornosti, impulzivita, terapie poúrazových stavů, kompenzace počáteční fáze demence atd.). Aplikace je dostupná zdarma, šířena pod svobodnou licencí. Pro užívání aplikace je však potřeba Microsoft Kinect sensor.

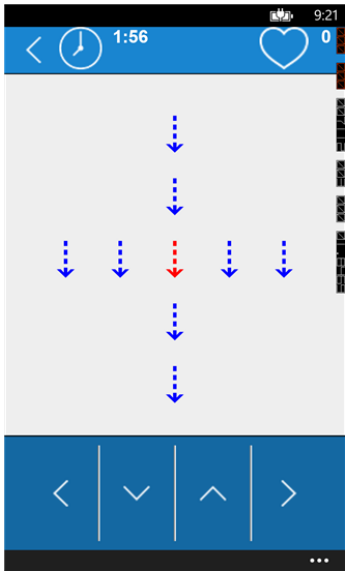


Obrázek 7: Newron - úkol ve stylu matematického hlavolamu hanojské věže

Obrázek 8: Newron - úkol vybarvování obrázků

3.2.2 Mind Games

Mind Games je typickým zástupcem volně dostupných aplikací pro rozvoj kognitivních schopností jedince. Nabízena je pro všechny tři mobilní platformy - iOS, Android i Windows Phone. Jedná se o nejlépe hodnocenou aplikaci tohoto zaměření na platformách Android a Windows Phone 8/8.1. Ve svých celkově 29 úkolech se zaměřuje zejména na paměť (vizuálně-prostorovou, pracovní), verbální koncepty a paměť (bohužel jen v anglickém jazyce), a dále i úzce specializované úkoly jako například schopnost vybavení si obličeje a podobně. Aplikace je volně dostupná, s tím že placená verze pouze odstraní reklamní sdělení. Na obrázku 9 lze vidět jednoduchost uživatelského rozhraní. Konkrétně se jedná o variantu flankerova úkolu pro rozvoj pozornosti a schopnosti filtrovat vizuální vjemy.



Obrázek 9: Mind games - úkol založený na principu Eriksen-Flankerova úkolu

3.3 Shrnutí aplikací

V následujících tabulkách je shrnuta rešerše dostupných aplikací pro rozvoj kognitivních schopností. Mimo výše popsaných aplikací byly vybrány průřezově aplikace zaměřující se na různé oblasti kognitivního rozvoje a to jak komerční, tak volně dostupné aplikace.

Oblasti rozvoje	Dostupnost	Výhody	Nevýhody
Název: *Program Activate (C8 Sciences)* **Web**: `http://www.c8sciences.com/`			
Pozornost, rychlost zpracování podnětu, paměť, pracovní paměť	Pouze placená verze, platforma PC a OSX	Navrženo pro cílovou skupinu děti školního věku, dostupné verze pro školství a pro domácí použití, komponenta fyzického cvičení spolu s programem pro dotyková zařízení	Neexistující česká jazyková mutace, placená verze
Název: *EDA Play* **Web**: `http://www.edaplay.cz/`			
Zrakové vady, koordinace ruka-oko, jemná motorika	Pouze placená verze, platforma OSX	Česká jazyková mutace, velmi jednoduché uživatelské rozhraní, rozsáhlá komunita	Pouze na platformě OSX, více méně zaměřené jen na kompenzaci zrakových vad
Název: *Brain Metrix* **Web**: `http://www.brainmetrix.com/`			
Paměť, reflexy, IQ, koncentrace, kreativita, matematika, pozornost, vizuálně-prostorový rozvoj	Platforma PC, zdarma, on-line či instalace	Široká nabídka různých her, jednoduché použití	Reklamy, přihlášení skrze třetí stranu
Název: *Cogmed (verze JM - předškolní věk, RM - školní věk, QM - dospělí)* **Web**: `http://www.cogmed.com/`			
Pozornost, koordinace ruka - oko, paměť, pracovní paměť	Platforma PC, placená verze	Individualizované plány pro jednotlivce, individuální koučing	Placené, opravdu komplexní program vyžadující konzultace, potréninková sezení atd.
Název: *Mind Games* **Web**: `http://www.mindware.mobi/blog/`			
Pozornost, vizuálně-prostorová paměť, pracovní paměť, flexibilita, verbální rozvoj (pouze anglický jazyk)	Android, iOS, Windows Phone 8/8.1, zdarma (+ reklamy)	Zdarma, široké spektrum rozvoje, 29 různých úkolů, jednoduché uživatelské rozhraní, mobilní platformy	Pouze anglické prostředí (pro české uživatele jsou tedy verbální úkoly nepoužitelné)

Tabulka 1: Shrnutí dostupných aplikací pro rozvoj kognitivních schopností

Oblasti rozvoje	Dostupnost	Výhody	Nevýhody
Název: *Brain Fitness (verze PRO, junior, special edition, impulse control)* **Web**: `http://www.mindsparke.com/`			
Paměť, soustředění, pružnost, IQ, kreativita, síla vůle, meditace, redukce vznětlivosti	Placená verze, platforma PC	Garance vrácení peněz, zákaznický servis	Mimo verzi junior není založeno na principu her
Název: *Newron* **Web**: `http://www.newron.cz/newron/`			
Primárně kompenzace u dětí s PAS, v budoucnu pracovní paměť, pozornost, kompenzace poúrazových stavů, obecný kognitivní rozvoj	Platforma PC, zdarma	České prostředí i vývojový tým, ovládání gesty prostřednictvím Microsoft Kinect	Nutnost využití Microsoft Kinect
Název: *Lumosity* **Web**: `http://www.lumosity.com/`			
Paměť, problem solving, flexibilita, pozornost, rychlost reakce	Webové rozhraní, aplikace pro Android i iOS, zdarma	Navrženo ve spolupráci s neurovědci, webové rozhraní i aplikace pro mobilní platformy, různé úrovně tréninku	Ve verzi zdarma málo aplikací, v podstatě nutnost dokoupit prémiovou verzi (mikrotransakce)
Název: *My Brain Trainer* **Web**: `https://www.mybraintrainer.com/`			
Reakční čas, schopnost vykonávací, krátkodobá paměť, vizuálně-prostorový trénink, pracovní paměť (n-back task)	Webové i desktopové rozhraní, placené (existuje trial verze)	48 různých úkolů, zpětná analýza výkonu, jasné instrukce, široké spektrum úkolů	Nepřehledné webové rozhraní, placené
Název: *Brain Experiment* **Web**: `http://www.brainexperiment.org/`			
Paměť, reakční čas, soustředění, vizuálně - prostorové zpracování, matematika	platforma PC, zdarma (s reklamami)	Jednoduchost použití, různorodé hry, zdarma sledování výsledků/pokroku	reklamy

Tabulka 2: Shrnutí dostupných aplikací pro rozvoj kognitivních schopností

4 Vybrané metody pro rozvoj kognitivních schopností

Pro samotnou vzorovou aplikaci byly na základě buďto průzkumu v oblasti metod rozvoje kognitivních schopností, či na základě vypozorování těchto metod u aplikací, které byly zkoumány v rešerši diplomové práce, vybráno několik konkrétních úkolů. Při výběru bylo přihlédnuto zejména na vhodnost implementace pro prostředí tabletu s dotykovým ovládáním, nepříliš velkou složitost uživatelského rozhraní úkolu a také na určité oblasti rozvoje. Aplikace ku příkladu není určena pro kompenzaci zrakových či jazykových poruch, ani se nezaměřuje na rozvoj komunikačních a sociálních schopností jedince. Zaměřením jednotlivých úkolů se aplikace snaží zejména o rozvoj pozornosti, pracovní paměti, fluidní inteligence, prostorové paměti a prostorové představivosti. Samotný fakt, že je aplikace uzpůsobena pro dotykové ovládání, a také koncepce uživatelského rozhraní přináší u úkolů rozvoj koordinace ruka - oko a jemné motoriky, jak již bylo zmíněno výše.

4.1 Úkol Diamanty

Prvním úkolem aplikace je úkol nazvaný "diamanty". V tomto cvičení je úkolem uživatele zopakovat posloupnost vysvicování (reprezentováno jako změna barvy) daného počtu "kamenů"rozmístěných v mřížce, s tím, že počet vysvícených kamenů se po každém kole zvětšuje o 1. To, které kameny budou vysvíceny je v každém kole náhodné. Jedná se o jednoduchou implementaci Corsi block-tapping testu (4.1.2), viz například podobně koncipovaný úkol v aplikaci Cogmed (3.1.3), ve kterém se u uživatele zjišťuje jeho schopnost zapamatovat si danou posloupnost vysvícení kamenů. Variantou by bylo vysvítit na určitou dobu (dynamicky se měnící) určitý počet kamenů najednou a poté po uživateli požadovat dotyk právě těchto kamenů (v libovolném pořadí). V aplikaci bude možnost nastavení jednotlivých parametrů cvičení (doba vysvícení kamene, počáteční počet vysvícených kamenů a podobně). Uživatel je na správný i špatný průběh úkolu upozorněn příslušným zvukovým i vizuálním znamením.

4.1.1 Návrh uživatelského rozhraní

Uživatelské rozhraní úkolu Diamanty je koncipováno následovně: v levé části obrazovky jsou rozmístěny v pravidelné mřížce jednotlivé diamanty, v pravé části obrazovky je poté umístěn obrázkový pomocník, indikující uživateli zda je v danou chvíli na tahu uživatel nebo "počítač". Dále nejsou na obrazovce žádné jiné viditelné prvky uživatelského rozhraní. Pro návrat do seznamu úkolu bude sloužit kontextová klávesa pro návrat operačního systému Android. Po provedení úkolu se uživateli vždy zobrazí výsledek (u úkolu Diamanty pouze výsledek správně/špatně) v grafické podobě uprostřed obrazovky a

zazní příslušné zvukové znamení. Pro pokračování v provádění úkolu uživatel pouze provede libovolné dotykové gesto na obrazovku. Uživatelské rozhraní tohoto typu aplikací by mělo být koncipováno co možná nejjednodušeji, s přihlédnutím k tomu, že cílovou skupinou mohou být například skupiny lidí s určitým druhem mentálních či zdravotních deficitů a ne jen naprosto zdraví jedinci. Z tohoto důvodu by příliš komplexní či nepřehledné uživatelské rozhraní mohlo mít za následek sníženou schopnost soustředit se na daný úkol, což by v důsledku mohlo znamenat i horší výsledky.

4.1.2 Corsi Block-tapping test

Corsiho Block-tapping test je psychologický test, který testuje vizuálně-prostorovou krátkodobou pracovní paměť. Jeho podstata spočívá v napodobování (zopakování) sekvence rozsvěcování až devíti (v některých aplikacích až 12) prostorově oddělených bloků. Obtížnost zopakování sekvence je zpočátku jednoduchá, obvykle je nejprve test proveden se dvěma bloky, ale s rostoucím počtem bloků v sekvenci obtížnost stoupá do té doby, než testovaný subjekt není schopen sekvenci správně zopakovat. Počet bloků, které je subjekt ještě schopen úspěšně zopakovat se nazývá tzv. Corsi Span (corsiho rozsah). Velikost tohoto rozsahu je u zdravého jedince průměrně 5.

Studie FMRI ukázala na testovaných subjektech, že se zvyšováním počtu bloků v sekvenci nedochází ke zvyšování obecné mozkové aktivity. Ze zvyšujícím se věkem jedince byla pozorována i zvyšující se hodnota Corsi span. Tato hodnota však kumuluje kolem 14 věku, kdy rozdíly oproti dospělým lidem nejsou signifikantní. Výzkum také ukázal, že v hodnotě Corsi span nejsou žádné genderové rozdíly.

Corsiho block tapping task je používán pro testování mnoha různých věcí, včetně ztráty paměti, testování pacientů s poškozením mozku, testování prostorové paměti a neverbální pracovní paměti. V dnešní době je již mnoho aplikacích pro rozvoj kognitivních schopností, jenž jsou na variantách tohoto původně psychologického testu založeny.

Zpětný Corsi block tapping je lehce pozměněnou verzí původního úkolu. Ve zpětné verzi je úkolem subjektu zopakovat sledovanou posloupnost v opačném pořadí než v jakém byla provedena. Ačkoliv formát normálního a zpětného Corsi block tetu je analogický dopřednému a zpětnému Digit span testu (který testuje verbální paměť namísto vizuálně prostorové), zpětný Corsiho block test se liší od zpětného Digit span testu relativní obtížností. Ve studii o souvislosti těchto dvou testů bylo zjištěno, že zatímco zpětná verze Digit span testu byla pro subjekty mnohem těžší než dopředná verze, mezi dvěma verzemi Corsi block testu nebyl zjištěn žádný významný rozdíl co se týče obtížnosti provedení a intenzity rozvoje kognitivních schopností.

Zpětný Corsi block tapping task byl také použit při zkoumání rozdílů mezi procesy, které mozek používá při vykonávání Digit span test (dopředná i zpětná varianta) a Corsi block tapping test (opět obě varianty). Tato studie proběhla na dvou skupinách dětí a to na kontrolní skupině zdravých dětí a na skupině dětí s deficitem ve vizuálně prostorovém učení. Bylo zjištěno, že výkon pouze druhé skupiny ve zpětné variantě Corsi Block testu byl výrazně horší než v jeho klasické variantě, zatímco výkon u zpětného Digit spanu byl v porovnání s klasickou variantou horší u obou testovaných skupin. Toto ukazuje, že zpětná varianta Corsiho block tapping testu využívá speciální prostorové procesy mozku. V aplikaci je použita pouze běžná varianta úkolu, nikoli tedy zpětná verze.

4.2 Krychle

V tomto cvičení je úkolem uživatele pomocí dotykových gest "táhnutí"zopakovat posloupnost předem provedenou "počítačem", v tomto případě jde o trojrozměrnou prostorovou informaci v podobě otáčení krychle (kde každá stěna krychle může být pro snadnější orientaci jinak zabarvena). Počet otočení kostky může být buďto předem nastaven, či se může dynamicky měnit v závislosti na výsledcích uživatele a směry jsou generovány náhodně v každém pokusu. Opět se jedná o variaci cvičení na krátkodobý rozsah paměti. V aplikaci lze nastavit parametry tohoto cvičení jako například rychlost otáčení, počet otočení a podobně. Uživatel je na správný i špatný průběh upozorněn příslušným zvukovým i vizuálním znamením. Cílem procvičování tohoto úkolu je jednak rozvoj koordinace ruka-oko, tak prostorové orientace a představivosti jedince, a dále také rozvoj krátkodobé paměti a tzv. memory spanu 4.2.2.

4.2.1 Návrh uživatelského rozhraní

Uživatelské rozhraní úkolu krychle je koncipováno velmi jednoduše - sestává pouze z krychle umístěné ve středu obrazovky (případně mírně posunuté do její levé části). Krychle má každou stěnu zabarvenou jinou barvou. V pravé části obrazovky je poté vizuální informace, indikující uživateli to, zda je na tahu on nebo počítač. Vizuální informace může být doplněna i zvukovým signálem.

4.2.2 Memory Span

V psychologii a neurovědě je Memory span nejdelší možná množina položek, kterou je jedinec schopen zopakovat ve správném pořadí ihned po tom co mu množina byla předvedena. Položkami mohou být slova, číslice či písmena. Tento task je znám jako Digit span

v případě položek odpovídajícím číslicím. Memory span je běžným ukazatelem krátkodobé paměti. Zpětný memory span je složitější variantou, jejíž podstatou je vybavení si položek v opačném pořadí než ve kterém byly představeny.

Funkcionálně představuje paměťový rozsah počet diskrétních jednotek, na které dokáže jedinec distribuovat svou pozornost a úspěšně je složit v jeden celek. Obecně tedy memory span odpovídá schopnosti jedince ihned po jediné prezentaci zreprodukovat sérii diskrétních stimulů v jejich původním pořadí. Jak již bylo zmíněno, pro tento účel může být použit prakticky jakýkoliv materiál, ať už číslice, písmena, slova, zvuky a jakýkoliv smyslový orgán nebo jejich kombinace.

4.3 Rotování

V tomto cvičení je úkolem správně vybrat ze dvou obrázků zobrazených v první půli obrazovky a to na základě referenčního útvaru zobrazeného v druhé půli obrazovky, přičemž oba útvary jsou proti referenčnímu natočeny o určitý úhel. Gestem dotyku na daný útvaru ho poté uživatel zvolí a dojde k vyhodnocení. Výsledek je opět uživateli oznámen jak textovou a obrázkovou informací, tak příslušným zvukovým signálem. Obtížnost je u takovéto aplikace dána zejména velikostí úhlu natočení a také vybraným referenčním útvarem.

4.3.1 Návrh uživatelského rozhraní

Uživatelské rozhraní úkolu diamanty sestává z oblasti v prostřední části obrazovky, ve které se zobrazují jednotlivé obrazce. V současné podobě se jedná o předměty běžného života (tužka, míč, kniha a podobně). Dále se v pravé části obrazovky nachází jednoduchý slovní popis úkonu, oznamující uživateli současnou velikost zátěžové hodnoty n. Pro výsledkovou část uživatelského rozhraní byla vyzkoušena varianta s emotikony označující úspěch (smějící se emotikon), případně neúspěch (mračící se emotikon). Vodorovné pruhy u každého emotikonu, jejich velikost a poměr určují úspěch, případně neúspěch v daném kole. Uživatel se tedy snaží mít co možná nejširší pruh u smějícího se emotikonu.

4.3.2 Mentální rotace

Mentální rotace je asociována se schopností prostorového vnímání, obecné inteligence a se schopností zpracování vizuální informace. Mentální rotace je v podstatě schopnost lidského mozku pohybovat či rotovat (pouze však imaginárně) s objekty. [12]

Mentální rotace může být rozčleněna na následující kognitivní fáze :

1. Vytvoření mentálního obrazu objektu

2. Rotování objektem do té doby, než je možno provést porovnání
3. Provedení porovnání
4. Rozhodnutí o tom, zda jsou objekty stejné
5. Oznámení rozhodnutí

V klasickém dvouprostorovém testu mentální rotace je subjekt požádán, aby porovnal dva objekty (geometrické tvary, písmena) a uvedl, zda se jedná o stejné objekty či o objekty zrcadlově převrácené. Běžně jsou objekty v testu pootočené o specifický úhel (například 60, 120, 180 stupňů). Některé z párů jsou stejné, pouze pootočené o jiný úhel, jiné jsou zrcadlově převrácené. Subjektu je zobrazena množina těchto párů a ohodnocuje se, jak přesně, a v jakém čase dokáží mezi těmito dvojicemi rozlišit.

4.4 Obrazce

U tohoto cvičení jsou uživateli zobrazovány v daném nastavitelném časovém intervalu jednotlivé obrazce (v současné verzi se jedná o ilustrace jednoduchých předmětů denního života), přičemž jeho úkolem je použít dotykové gesto na obrazec v případě, že se daný obrazec ve zobrazované posloupnosti již objevil a to přesně před n kroky, kde n je taktéž nastavitelné. Implementace tohoto úkolu odpovídá tzv. n-back úkolu (4.4.2). Uživatel je na špatnou i správnou volbu ihned upozorněn příslušným zvukovým znamením (i pokud dotykové gesto nepoužije tehdy, kdy měl).

4.4.1 Návrh uživatelského rozhraní

U úkolu obrazce je opět uživatelské rozhraní koncipováno velmi jednoduše. Sestává ze střední části, ve které se zobrazují jednotlivé obrazce v pravidelných časových intervalech a poté vizuální indikace v pravé části obrazovky, připomínající uživateli zátěžový index n.

4.4.2 n-back úkol

U úkolu n-back je subjektu prezentována sekvence podnětů, a jeho úkolem je indikovat kdy současný podnět odpovídá podnětu před právě n kroky v prezentované sekvenci. Faktor zatížení n může být upraven a z jeho velikosti se poté odvíjí složitost úkolu. Hovoříme poté konkrétně o 1-back, 2-back, 3-back a podobně. Podnět u úkolu n-back nemusí být pouze v podobě vizuální (v případě vizuálního n-back), ale například i zvukové (auditivní n-back), případně využívající jiné smysly.

Pro objasnění, vizuální zkouška n-back je podobná klasické hře pexeso. Avšak místo fixních obrazců rozmístěných pevně po hrací ploše by zde byl jen jeden obrazec, který by se objevoval v různých pozicích na hrací ploše při každém kole. 1-N znamená, že si subjekt musí pamatovat pozici položky jedno kolo zpět. 2-N znamená, že si subjekt musí pamatovat pozici položky dvě kola zpět a tak dále.

Například, auditivní podoba úkolu 3-back by se mohla skládat z prezentace subjektu z následujícího seznamu písmen:

T L H C H O **C** Q L **C** K **L** H C Q T R R K C H R

Subjekt má poté indikovat prezentaci pouze podtrženě vyznačených písmen, jelikož ty odpovídají prezentaci těchto písmen před právě třemi kroky.

Úkol n-back stimuluje aktivní část pracovní paměti jedince. Pokud je zátěžová hodnota **n** rovna 2 a více, nestačí subjektu pouze zachovat v paměti reprezentaci nedávno prezentovaných položek. Pomyslný "buffer"pracovní paměti si musí jedinec neustále obnovovat na základě prezentované informace, aby mohl vyhodnotit se kterým podnětem má být momentálně prezentovaný podnět porovnán. Aby tohoto dosáhl, musí jedinec jak uchovat, tak i manipulovat s informací v pracovní paměti.

4.4.3 Duální n-back

Duální varianta úkolu n-back je variantou kdy jsou subjektu prezentovány 2 různé zdroje podnětů zároveň - například vizuální n-back spolu s auditivním n-back. V aplikaci je použita pouze základní verze úkolu n-back, pro případné rozšíření na duální variantu jsou však ve frameworku aplikace potřebné konstrukty.

4.4.4 Použití n-back úkolu v praxi

Úkol n-back je v dnešní době široce využíván mimo experimentální, klinické a lékařské prostředí. Společnosti zabývající se vzděláváním používají různé verze tohoto úkolu (ve spojení s jinými metodami) pro ostentativní zlepšení fluidní inteligence klientů. Dále je tento úkol používán psychology pro zlepšení schopnosti soustředění jedinců s poruchou pozornosti s hyperaktivitou (ADHD) a pro rehabilitaci jedinců, kteří utrpěli traumatické poranění mozku. Experimenty ukázaly, že u participantů, kteří pravidelně používali tento úkol, se zlepšila schopnost pozornosti až po osm měsíců po ukončení tréninku. Stále však ještě přetrvává skepse nad tím, zda trénink s využitím n-back úkolu, případně podobných úkolů pro rozvinutí kognitivních schopností má užitek i z dlouhodobějšího

hlediska, či zda jsou efekty tréninku pouze přechodného charakteru a také to, zda účinky tréninku lze vysledovat i v obecnější rovině kognitivních schopností, například u fluidní inteligence. V dnešní době, a to navzdory tvrzením firem, které komerčně produkty pro zlepšení kognitivních schopností nabízejí, je v literatuře nedostatečné množství důkazů potvrzujících to, že tréninkem účinnosti pracovní paměti by se rozvinuly i obecnější složky kognitivního fungování jedince.

5 Implementace vybraných metod pro aplikaci operačního systému Android

Jak již bylo zmíněno, došlo v rámci diplomové práce k implementaci vzorové aplikace, obsahující celkově čtyři úkoly zaměřující se na různé složky kognitivních schopností jedince. Aplikace je určena pouze pro operační systém Android, tudíž je implementována s pomocí Android Software Developement Kit (SDK). Primární, a v současné době jediné podporované ovládání aplikace je pomocí dotykových gest, které operační systém nativně obsluhuje.

5.1 Operační systém Android

Operační systém Android je v dnešní době nejrozšířenějším a nejpoužívanějším operačním systémem pro mobilní zařízení navrženým primárně pro dotyková zařízení. Prvotně byl vyvinut pro mobilní telefony, tzv. "smartphone"a tablety, dnes však existují i specializovaná uživatelská rozhraní pro televizní zařízení (Android TV), automobily (Android Auto) a různé druhy zařízení nositelné elektroniky (Android Wear). Jedná se o operační systém postavený na monolitickém jádře Linux. Podle údajů z února 2015 je v aplikačním obchodě firmy Google (Google Play store), přibližně 1 400 000 různých aplikací. Od roku 2015 má díky obrovskému rozšíření mobilních zařízení tento operační systém největší uživatelskou základnu ze všech běžných operačních systému (včetně desktopových systému, jako např. Microsoft Windows). Jen v roce 2014 bylo 48.61% všech prodaných zařízení (tablety, laptopy, osobní PC, ostatní zařízení) založeno právě na Androidu. Není tedy pochyb o tom, že systém je dnes velmi rozšířen.

5.1.1 Vývoj aplikací pro operační systém Android

Důležitým faktem vývoje pro operační systém Android je to, že při vývoji musíme specifikovat, pro kterou verzi je aplikace určena, na základě čehož se využije příslušné API (knihovny v Android SDK). Aplikace nemusí být zpětně kompatibilní se staršími verzemi než na kterou byla zacílena, dopředná kompatibilita je však zajištěna - u aplikací se tedy specifikuje pouze minimální verze API, na kterou je cílena. V současnosti je na trhu již verze 5.1 (kódové označení "Lollipop"), vydaná v březnu 2015. Vzorová aplikace této diplomové práce však bude specifikována pro minimální API verze 15, což odpovídá verzi operačního systému 4.0.3 (kódové označení Ice Cream Sandwich). Důvodem minimálního API verze 15 je jednak nekompletnost Java napojení na knihovnu OpenGL ES 2.0 u nižších verzí, ale hlavně nízké rozšíření nižších verzí mezi spotřebiteli (dle údajů přístupů

zařízení na aplikační obchod Google Play store). Bylo by tedy poměrně nepodstatné brát v úvahu vývoj pro tyto verze. Zařízení běžící na verzi 4.0.3 a výše, tvoří přibližně 94% všech zařízení s operačním systémem Android.

V tabulce 3 je uveden souhrn různých verzí, jejich kódová označení, datum vydání, API level pro vývoj a jejich procentuální rozšíření mezi zařízeními spotřebitelů (data k 6. březnu 2015).

Verze	**Kódové označení**	**Datum vydání**	**API level**	**Procentuální zastoupení**
5.1.x	Lollipop	Březen, 2015	22	0.4%
5.0.0–5.0.2		Listopad, 2014	21	5.0%
4.4.0–4.4.4	KitKat	Říjen, 2013	19	41.4%
4.3.x	Jelly Bean	Červenec, 2013	18	5.6%
4.2.x		Listopad, 2012	17	18.6%
4.1.x		Červenec, 2012	16	16.5%
4.0.3–4.0.4	Ice Cream Sandwich	Prosinec, 2011	15	5.7%
2.3.3–2.3.7	Gingerbread	Únor, 2011	9	6.4%
2.2	Froyo	Květen, 2010	8	0.4%

Tabulka 3: Verze operačního systému Android, zdroj: [13]

Aplikace pro Android jsou vyvíjeny primárně v jazyce Java, použitím Android Software Developement Kit (Android SDK), který v sobě zarhnuje různé komponenty a nástroje jako debugger, knihovny jednotlivých API operačního systému pro jazyk Java, nástroj pro překlad výsledných souborů sestavení do formátu DEX a podobně. Aplikace vyvinuté v jazyce Java jsou poté spuštěny ve virtuálním běhovém prostředí Dalvik Virtual Machine (DVM), jenž je obdobou klasického Java Virtual Machine (JVM). Stejně jako JVM podporuje Just In Time kompilaci Java byte-kódu a obsahuje automatický hlídač alokace paměti (garbage collector). Oproti JVM je však upraven, je postaven na registrovém modelu virtuálního stroje (na rozdíl od zásobníkového modelu u klasického JVM). V praxi to znamená mírně rychlejší běh vykonávání instrukcí a další možné optimalizace, které byly vyžadovány kvůli omezenému výkonu mobilních zařízení. Jádro Linux pro každý spuštěný proces operačního systému rezervuje vlastní instanci virtuálního stroje DVM

a to z důvodu odstranění závislostí jednotlivých aplikací a virtuálních strojů, kdy pád jednoho virtuálního stroje by mohl způsobit pád více aplikací. Pro samotné aplikace je využívána taktéž upravená verze Java byte-kódu v podobě formátu DEX, který odstraňuje některé nedostatky klasického .class formátu, přičemž aplikace jsou baleny do jednotného balíčku formátu APK (obdoba JAR formátu u JVM). U nejnovějších verzí operačního systému (počínaje verzí s kódový označením KitKat) bylo nasazeno nové virtuální běhové prostředí s názvem Android runtime (ART). Jedná se o vylepšenou verzi běhového prostředí DVM, s podporou spouštění stejného souboru byte-kódu (DEX). Oproti DVM však došlo k několika vylepšením, z nichž nejpodstatnější je Ahead Of Time kompilace byte-kódu do nativního kódu (na rozdíl od Just In Time kompilace u DVM - byte-kód je v případě ART překládán do nativního kódu již během instalace aplikačního balíčku) a vylepšený garbage collector. Ve výsledku se toto nové běhové prostředí může projevit podstatným zlepšením výkonu aplikace, minimálně v rychlosti spuštění aplikace. [15] [14]

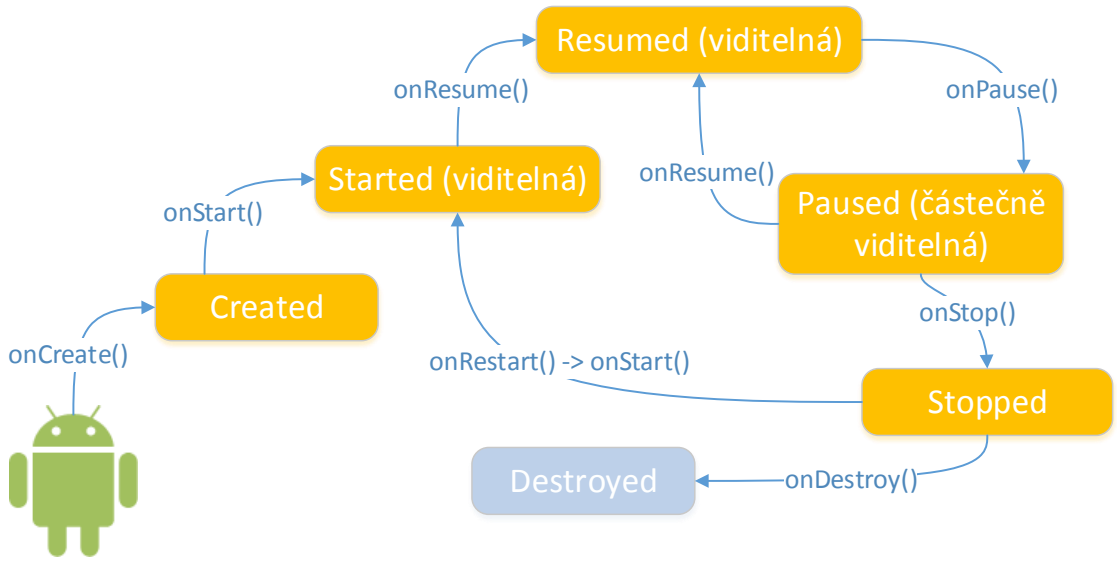
5.2 Komponenta Activity

Komponenta Activity je jednou ze čtyř základních komponent aplikace v operačním systému Android (spolu s komponentami Service, Content provider a Broadcast receiver), reprezentující jednu ”obrazovku”aplikace s vlastním uživatelským rozhraním, nezávislým na ostatních instancích komponenty Activity. V aplikaci se implementuje zděděním z třídy Activity a implementováním potřebných metod, které operační systém volá při přechodu Activity mezi jednotlivými stavy jejího životního cyklu. Uživatelské rozhraní Activity je poskytnuto hierarchií tzv. views, tedy objekty dědící ze třídy View. Android SDK pochopitelně poskytuje mnohá základní *view*, jako například různá tlačítka, textová pole, kontejnery ostatních *view* sloužící pro správné a responzivní pozicování komponent a různé další prvky uživatelského rozhraní. Uživatel zděděním třídy View a implementací potřebných metod může implementovat vlastní prvky uživatelského rozhraní pro použití v aplikaci [14].

Pro aplikaci bude komponenta Activity použita jako komponenta poskytující uživatelské rozhraní pro hlavní menu, pro obrazovku nastavení a také pro jednotlivé úkoly aplikace, jelikož tento systém nám aplikaci dobře architektonicky rozdělí a zajistí nezávislost jednotlivých částí.

5.2.1 Životní cyklus komponenty Activity

Veškeré instance komponenty Activity jsou interně v operačním systému spravovány ve struktuře založené na principu zásobníku, tzv. *activity stack*. Pokud je spuštěna nová instance komponenty Activity, je uložena na vrchol zásobníku a přenesena do popředí. Ve chvíli kdy se instance nachází v popředí a je na ní uživatelský focus, nachází se ve stavu *running*. Pokud instance ztratila focus uživatele, ale stále je viditelná (byla tedy překryta částečně viditelnou či transparentní jinou instancí Activity), nachází se ve stavu *paused*. Instance ve stavu *paused* má uchovány veškeré informace o stavu svého uživatelského rozhraní, může však být v situacích, kdy je volná paměť systému kriticky malá, operačním systémem ukončena. Pokud je instance kompletně překryta jinou instancí komponenty Activity (tato nová instance je tedy na vrcholu *activity stack* zásobníku), přejde do stavu *stopped*. V tomto stavu je nadále zachován její kompletní stav (včetně stavu uživatelského rozhraní, a tedy i veškerých členských proměnných). Instance ve stavu *stopped* může být opět operačním systémem ukončena při potřebě zejména operační paměti pro jiné aktivní instance. Instance nacházející se ve stavu *paused* nebo *stopped* může být tedy na požadavek operačního systému ukončena, čímž dojde ke ztrátě veškerých informací o stavu členských proměnných a uživatelského rozhraní. Pro lepší ilustraci lze vidět životní cyklus na obrázku 10.



Obrázek 10: Životní cyklus instance komponenty Activity

(zdroj: http://developer.android.com/reference/android/app/Activity.html)

5.2.2 Android Manifest

Při použití libovolné komponenty aplikace (viz výše) musí být tato komponenta deklarovaná v souboru *AndroidManifest.xml*, který musí být umístěn v kořenové složce projektu.

```
<?xml version="1.0" encoding="utf−8"?>
<manifest ... >
    <application android:icon="@drawable/app_icon.png" ... >
        < activity  android:name="com.example.project.ExampleActivity"
                    android:label="@string/example_label" ... >
        </ activity >
        ...
    </application>
</manifest>
```

Výpis 1: Příklad deklarace komponenty Activity v souboru aplikačního manifestu

Kromě nutnosti deklarace použití jednotlivých komponent nám *manifest* soubor umožňuje definovat různé požadavky, které aplikace na klientský systém klade, jako například:

- Uživatelská povolení, která aplikace vyžaduje, ku příkladu připojení k internetu, čtení kontaktů v přístroji uživatele apod.
- Deklarace minimální API úrovně operačního systému uživatele
- Deklarace použití hardwarových součástí zařízení (například fotoaparát, bluetooth zařízení)
- Některé knihovny, se kterými má být sestavení aplikace navázáno
- Deklarace různých dalších požadavků

Pomocí manifest souboru můžeme deklarovat požadovanou minimální verzi standardu OpenGL ES, kterou zařízení implementuje.

```
<uses−feature android:glEsVersion="0x00020000" android:required="true" />
```

Výpis 2: Deklarace minimální verze OpenGL ES zařízení v souboru aplikačního manifestu

5.3 OpenGL ES v operačním systému Android

OpenGL ES (OpenGL for **E**mbedded **S**ystems) je podmnožina standardu OpenGL, specifikujícího aplikační rozhraní (API) pro zobrazování 2D a 3D počítačové grafiky. Je navrženo speciálně pro použití v operačních systémech mobilních telefonů, tabletů a podobných

mobilních zařízení a v některých herních konzolích jako "lightweight"varianta klasického OpenGL standardu. Oproti němu nabízí jasněji a jednodušeji definované API, menší paměťový otisk a díky dalším úsporám i nižší energetickou náročnost, která je u přenosných zařízení v dnešní době stále podstatným kritériem

5.4 Vykreslování 2D a 3D grafiky pomocí OpenGL ES 2.0

Grafický standard OpenGL ES je svou architekturou ve své podstatě stavovým automatem. Na začátku každého vykreslování je tedy uvedení grafického kontextu do požadovaného stavu. Stavem se rozumí veškeré navázání vrcholových dat, texturových dat, programů vertex a fragment shaderu a dalšího nastavení interního stavu OpenGL ES kontextu.

Vykreslování grafických primitiv na obrazovku začíná množinou jednoho nebo více vertex bufferů, které jsou naplněny poli vertex atributů. Tyto atributy jsou použity jako vstupy pro podprogram *vertex shader*. Běžnými vertex atributy jsou pozice vrcholu ve 3D prostoru, pozice texturových koordinát, které mapují obrazová data z jedné či více textur na vrchol, či koordináty normál, které jsou poté použity například při vypočtení odrazu světla, případně u dalších operací. Množina vertex bufferů, které dodávají data do jedné tzv. vykreslovací úlohy (rendering job, tedy kompletní proces vykreslení grafických dat na obrazovku) je nazývána *vertex array*.

Bližší pohled na použití grafického standardu OpenGL ES pro mobilní platformy lze najít například v bakalářské práci [18].

Na obrázku 11 je znázorněna zjednodušená ilustrace jedné vykreslovací úlohy.

Obrázek 11: Příklad vykreslovací úlohy OpenGL ES 2.0

(zdroj: http://duriansoftware.com/joe/An-intro-to-modern-OpenGL.-Chapter-1:-The-Graphics-Pipeline.html)

5.4.1 GLSurfaceView a Renderer

API operačního systému Android nám poskytuje dva konstrukty, které podstatně ulehčují práci s použitím implementace grafického standardu OpenGL ES. První z nich je třída GLSurfaceView, dědící z hierarchie třídy View, tedy, jak již bylo zmíněno, třídy používané pro tvorbu prvků uživatelského rozhraní. Třída GLSurfaceView nám poskytuje následující funkce:

- stará se o *"surface"*, což je v koncepci operačního systému Android speciální místo paměti, jenž může být zakomponováno do struktury View objektů,
- poskytuje získání kontextu OpenGL ES pomocí knihovny EGL, což umožňuje propojení knihovny OpenGL ES a již zmíněného *surface*, což ve svém důsledku znamená umožnění vykreslování grafického výstupu přímo na *surface*,
- má v kompetenci vykreslování z OpenGL ES kontextu na samostatném vlákně, čímž umožňuje oddělení vykreslování z grafického kontextu OpenGL ES od hlavního vlákna aplikace (které se například stará o vykreslování zbytku uživatelského rozhraní),
- poskytuje funkci mnohonásobného bufferingu (implicitně se jedná u většiny zařízení a implementací o triple-buffering),
- poskytuje fixní obnovovací dobu vykreslování (v závislosti na konkrétní implementaci je to 30 nebo 60 snímků za sekundu),
- implementováním metody *onTouchEvent(MotionEvent)* nám umožňuje reagovat na uživatelský vstup dotykovými gesty,
- použitím metod *onPause()* a *onResume()*, ovlivňující vykreslování na vlákně OpenGL ES kontextu, umožňuje reagovat na životní cyklus třídy Activity, ke které je každá instance přidružena.

Použití třídy *GLSurfaceView* (vytvořením vlastní třídy dědící z této třídy) odstíní uživatele od nízkoúrovňových konstruktů nutných pro získání OpenGL ES kontextu a pro jeho napojení na standardní zobrazovací manažer operačního systému.

Statické rozhraní *GLSurfaceView.Renderer* je rozhraním, jenž nám umožňuje implementovat metody grafického výstupu. Rozhraní obsahuje následující tři veřejné abstraktní metody, které jsou volány příslušnou třídou dědící ze třídy *GLSurfaceView*:

- *onDrawFrame(GL10 gl)* - metoda volána pro vykreslení současného interního stavu OpenGL ES. Jak již bylo zmíněno, metoda je v závislosti na implementaci volána přibližně 30 krát respektive 60 krát za sekundu. Vstupní parametr typu *GL10* je zde pouze z hlediska kompatibility a odpovídá rozhraní grafického standardu OpenGL ES verze 1.0. Pro použití metod standardu OpenGL ES 2.0 se používá statických metod a konstant třídy *GLES20*.
- *onSurfaceCreated(GL10 gl, EGLConfig config)* - metoda volána vždy když je OpenGL ES kontext vytvořen a ztracen (například při návratu zařízení ze spánkového režimu).
- *onSurfaceChanged(GL10 gl, int width, int height)* - metoda je volána vždy po volání metody *onSurfaceCreated* a při změně rozměrů surface (ku příkladu při otočení displeje, a tedy změně jeho orientace). Z parametrů *width* a *height* lze vyčíst současné rozměry zobrazovací části surface.

Typický příklad užití *GLSurfaceView* a s ním souvisejícího rozhraní *GLSurfaceView.Renderer* v komponentě typu Activity jako prvku pro zobrazení grafického výstupu standardu OpenGL ES je znázorněn na následujícím výpisu kódu:

```
public class MyGLRenderer implements GLSurfaceView.Renderer {

    public void onSurfaceCreated(GL10 unused, EGLConfig config) {...}

    public void onDrawFrame(GL10 unused) {...}

    public void onSurfaceChanged(GL10 unused, int width, int height) {...}
}

 ...

public class OpenGLES20Activity extends Activity {

    private GLSurfaceView mGLView;

    @Override
    public void onCreate(Bundle savedInstanceState) {
```

```
        super.onCreate(savedInstanceState);
        mGLView = new MyGLSurfaceView(this);
        setContentView(mGLView);
    }
    class MyGLSurfaceView extends GLSurfaceView {
        private final MyGLRenderer mRenderer;

      public MyGLSurfaceView(Context context){
            super(context);

            setEGLContextClientVersion(2);

            mRenderer = new MyGLRenderer();

            setRenderer(mRenderer);
        }
    }
}
```

Výpis 3: Použití třídy *GLSurfaceView* a rozhraní *GLSurfaceView.Renderer* pro získání grafického kontextu OpenGL ES

5.4.2 Uživatelský vstup

Implementací metody onTouchEvent(MotionEvent) třídy dědící ze třídy *GLSurfaceView* můžeme patřičně reagovat na uživatelský vstup. Jak již bylo zmíněno, grafický kontext standardu OpenGL ES (reprezentovaný pomocí třídy implementující rozhraní *GLSurfaceView.Renderer*) běží na samostatném vlákně, odděleném od hlavního vlákna uživatelského prostředí aplikace. Jelikož však třída *GLSurfaceView* běží právě na hlavním vlákně uživatelského prostředí aplikace, je pro mezi-vláknovou komunikaci využita metoda *queueEvent(Runnable r)* třídy *GLSurfaceView*, které je jako parametr předána anonymní třída rozhraní *Runnable*, jejíž metoda *run()* je spuštěna již na vlákně OpenGL ES kontextu. V metodě *run()* tedy můžeme například předat informace o proběhlé události dotykového gesta a na tuto událost patřičně reagovat ve třídě implementující rozhraní *GLSurfaceView.Renderer*, tedy přímo ve třídě zajišťující grafický výstup.

```
class MyGLSurfaceView extends GLSurfaceView {
    private MyRenderer mMyRenderer;
     ....
    public boolean onTouchEvent(MotionEvent event) {
            queueEvent(new Runnable() {
                public void run() {
```

```
                    mMyRenderer.handleInput(event);
                }});
            return true;
        }
    }
```

Výpis 4: Reakce na uživatelský dotykový vstup pomocí metody *onTouchEvent* třídy *View*

5.5 Existující frameworky pro tvorbu interaktivních grafických aplikací

Vývoj interaktivních grafických aplikací, kterou vzorová aplikace diplomové práce zajisté bude, je poměrně rozsáhlým implementačním problémem. Pro zjednodušení práce s grafickou knihovnou OpenGL ES, ale i s různými dalšími aspekty tvorby aplikace (jakými jsou například ozvučení, zpracování uživatelského vstupu, načítání externích dat a podobně) bylo již vyvinuto nepřeberné množství různých aplikačních frameworků. Jelikož však frameworky obecně poskytují buďto příliš komplexní řešení nebo naopak příliš specializované řešení nevhodné pro vyvíjený typ aplikace, popřípadě jsou z hlediska licencování nevhodné pro podobný projekt, byl vyvinut jednoduchý framework pro zjednodušení tvorby aplikace podobného zaměření.

5.5.1 Dostupné frameworky

Komerční i volně dostupné frameworky poskytují mnohdy opravdu komplexní multiplatformní řešení vývoje multimediální aplikace. V některých případech poskytují i vývojářské nástroje pro tvorbu aplikací, integrované vývojové prostředí a rozsáhlé knihovny obalující veškeré komponenty aplikace. Primárně bývají vyvíjeny pro tvorbu interaktivních her či aplikací s podobným zaměřením. Nejznámějšími zástupci komerčně dostupných frameworků jsou například framework Unity3D[1] (2D, 3D, multiplatformní, vlastní vývojářské nástroje a integrované vývojářské prostředí, licenční poplatky pokud je aplikace uveřejněna v aplikačním obchode Google Play Store) a Unreal Engine[2] (2D, 3D, multiplatformní, vlastní vývojářské nástroje a integrované vývojářské prostředí). Mezi nejpoužívanější volně dostupné frameworky se poté řadí zejména framework Cocos2D [3] (multiplatformní, volně dostupný, optimalizován zejména na vykreslování 2D grafiky), případně framework Libgdx [4] (multiplatformní, 2D, 3D, volně dostupné i zdrojové soubory). Hlubší souhrn dalších frameworků spolu s krátkým popisem a odkazy na přísl-

[1] Unity 3D - https://unity3d.com/
[2] Unreal Engine - https://www.unrealengine.com/
[3] Cocos2D - http://www.cocos2d-x.org/
[4] Libgdx - https://github.com/libgdx/libgdx/

šné weby je k dispozici například na webu `https://software.intel.com/en-us/android/blogs/2012/03/13/game-engines-for-android`.

5.6 Vlastní framework pro tvorbu aplikací

Jak již bylo zmíněno v úvodu sekce, pro zjednodušení práce jak s grafickým standardem OpenGL ES, pomocí něhož je zobrazován grafický výstup aplikace, tak s různými aspekty tvorby multimediální aplikace (jako například přehrávání zvuku, souborový vstup/výstup a podobně) byl implementován jednoduchý framework. Při implementaci bylo dbáno na rozdělení do jednotlivých logických sekcí, kde každá je zaměřená na určitý aspekt tvorby aplikace. Ve své implementaci byl tedy framework rozčleněn na 4 části:

- **Graphics** - část obalující práci s grafickým standardem OpenGL ES, přímo cílená na zjednodušení vykreslování grafických entit na scénu, tedy odstiňující uživatele od nízkoúrovňových konstruktů OpenGL API. V rámci této části byly připraveny 3 základní entity, jež v podstatě plně tvoří uživatelské rozhraní jednotlivých úkolů aplikace. Těmito entitami jsou entita *Sprite* (usnadňující zobrazování 2D obrázků na scénu, jejich následné pozicování, otáčení, ovládání velikosti), entita *Model* (usnadňující práci s 3D modely, poskytující podobně jako *Sprite* jejich pozicování, otáčení, ovládání velikosti a podobně) a entita *Text* (poskytující snadné zobrazení textové informace). V rámci této části byly také implementovány určité optimalizační kroky, zajišťující plynulejší chod aplikace i při zobrazování většího počtu geometrie, viz 5.6.3 5.6.4.

- **IO** - část frameworku, jejíž kompetencí je zejména načítání vrcholových dat ze souboru typu OBJ, běžně používaného pro jejich reprezentaci, a jejich následná reprezentace v paměti pro použití entitou *Model* a dále také usnadnění načtení souborů přiložených k sestavení aplikace (například různých textových souborů).

- **Sound** - část frameworku zjednodušující práci s konstrukty API operačního systému Android používanými pro přehrávání zvuků v aplikaci.

- **Utils** - jak již z názvu vypovídá, jedná se o část frameworku zaměřenou na různé další potřeby a aspekty tvorby multimediální aplikace. Nachází se zde ku příklady třídy pro reprezentaci vektorů (2D i 3D), třída pro usnadnění práce s maticovými operacemi a dále například třída usnadňující práci se sestavením a kompilací shader podprogramů pro další použití ve frameworku.

5.6.1 Diagram balíčků

V přílohách 19, 20, 21 a 22 jsou pro lepší znázornění rozdělení frameworku do logických celků uvedeny diagramy balíčků jednotlivých částí.

5.6.2 Práce s texturami

Pro abstrakci práce s grafickými texturami byla ve frameworku vytvořena třída *TextureHandler*, poskytující rozhraní k načtení textury z externích souborů přidružených k sestavení aplikace (resource soubory) a její následné navázání na texturovací jednotku grafického kontextu OpenGL. Třída je poté využívána pro již zmíněné základní grafické entity frameworku - Model, Sprite a Text. Na následujícím úryvku kódu je zobrazena práce se třídou *TextureHandler*, konkrétně načtení obrázku z resource souborů aplikace a jeho následné použití jako textury pro grafickou entitu Model:

```
public void onSurfaceChanged(GL10 gl, int width, int height)
{
    ...
    TextureHandler handler1 = new TextureHandler(context,                R.drawable.
        cubetextureflip,
        TextureHandler.TextureFiltering.linear);
    ...
    model.setTextureHandler(handler1);
    ...
}
```

Výpis 5: Ukázka použití třídy TextureHandler pro načtení textury z resource souboru

5.6.3 Sprite batching

Jak již bylo zmíněno, jednou ze základních entit poskytnutých frameworkem, sloužící pro zobrazení dvouprostorového obrazu je entita Sprite. Z pohledu optimalizace vykreslování by však nebylo výhodné vykreslovat každou instanci entity Sprite vlastním vykreslovacím kódem (tedy pro každou instanci vytvářet nový stav knihovny OpenGL a ten následně vykreslit), ale mnohem lepším řešením je vykreslit veškeré instance jednou vykreslovací úlohou. Pro řešení tohoto problému, obecně nazývaného jako sprite-batching, byla ve frameworku implementována třída SpriteBatcher, jež se stará o vykreslení veškerých instancí třídy Sprite v jedné vykreslovací úloze. Jejím úkolem je tedy sestavit pole vrcholových dat spolu s příslušnými atributy (UV koordináty, případně normály) a tyto poté připravit pro vykreslení.

5.6.4 Texture atlasing

Další optimalizační technikou použitou ve frameworku je tzv. texture atlasing. Jedná se o techniku nahrazení několika použitých textur jednou texturou kombinující všechny textury. Jelikož veškeré použité textury aplikace jsou známy již před sestavením, byl pro účely aplikace použit externí volně dostupný program generující z množiny zvolených vstupních textur výslednou texturu a k ní přidružený soubor obsahující informace o výsledných koordinátech původních textur v nově vzniklé textuře. Alternativním přístupem by bylo například generovat texturový atlas za běhu programu, pochopitelně však s určitou běhovou cenou.

5.6.5 Zobrazování textu

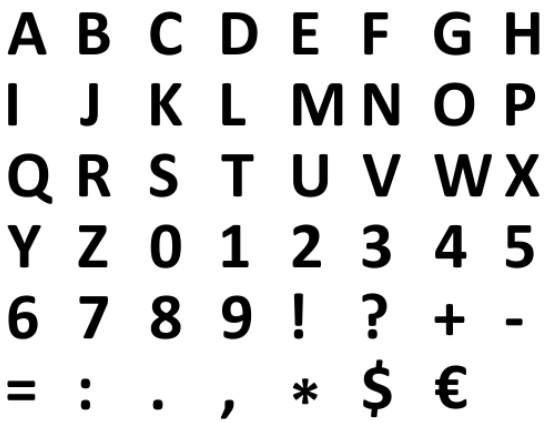
Při zobrazování textu v grafických aplikacích operačního systému Android využívajících OpenGL ES kontextu existuje několik možných přístupů. Standard OpenGL ES sám o sobě neposkytuje uživateli žádné nativní konstrukty k přímému zobrazení textu, tento nedostatek se tedy obchází vykreslováním textu podobným principem jako vykreslováním běžného objektu typu *sprite*, tedy použitím textury. Naivním přístupem by tedy bylo vytvořit veškeré možné texty jako textury a ty následně v aplikaci zobrazovat. V aplikaci se však často musí zobrazovat dynamický obsah a tak tento přístup není možný. Může být využit princip generování textury pomocí třídy *Canvas*, což je třída základního API operačního systému určená pro zobrazování grafického výstupu. Na následující ukázce lze vidět postup generování textové textury použitím třídy *Canvas* za běhu aplikace:

```
Bitmap bitmap = Bitmap.createBitmap(256, 256, Bitmap.Config.ARGB_4444);

Canvas canvas = new Canvas(bitmap);
bitmap.eraseColor(0);

Paint textPaint = new Paint();
textPaint.setTextSize(32);
textPaint.setAntiAlias(true);
textPaint.setARGB(0xff, 0x00, 0x00, 0x00);

canvas.drawText("Example_text", 16,112, textPaint);
```

Výpis 6: Ukázka generování textové textury pomocí třídy *Canvas*

Generování textury touto cestou je však z hlediska běhové náročnosti podstatně neoptimalizované, ku příkladu při zobrazování velmi dynamických dat by muselo docházet

v každém kroku běhu aplikace k vygenerování textury, jejímu následnému navázání na texturovací jednotku OpenGL kontextu a ke zobrazení. Při větším počtu zobrazovaných textů by to mohlo znamenat v konečném případě zhoršení obnovovací doby vykreslování, zejména u mobilních platforem, jenž neposkytují dostatečný výkon. Pro účely vykreslování textu tedy byla ve frameworku použita technika založena na již zmíněném texture atlasingu (5.6.4), kdy veškeré použité znaky jsou sestaveny do jedné textury a následně se z této textury pouze pomocí UV koordinát sestavuje výsledný textový řetězec. V praxi to znamená stálé navázání jedné před-generované textury na jednu texturovací jednotku grafického kontextu OpenGL, odpadnutí běhová cena generování textury pomocí třídy *Canvas*, ale pochopitelně i ztrátu flexibility, kterou nám třída Canvas při generování textu poskytuje. Na obrázku 12 lze vidět příklad výsledné textury používané pro zobrazování textu pomocí techniky texture atlasing.



Obrázek 12: Textura použita pro zobrazování textu technikou texture atlasing

5.6.6 Načítání OBJ souborů

Jelikož navržený aplikační framework má umožňovat jednoduché zobrazování 3D objektů na scéně, bylo potřeba implementovat konstrukty pro jednoduché načtení reprezentace těchto objektů do paměti z externího souboru. Jak již bylo zmíněno, grafická primitiva jsou v knihovně OpenGL reprezentována výčtem pozic vrcholů v prostoru a jejich případných atributů (UV koordináty, normály a podobně). Pro reprezentaci tohoto typu dat postačuje souborový typ *OBJ*. Souborový typ OBJ je jednoduchým, otevřeným datovým souborovým typem reprezentující 3D geometrii v textovém souboru vyvinutý společností Wavefront Technologies. V dnešní době jej využívá naprostá většina grafických aplikací pro tvorbu grafických scén a umožňuje do něj exportovat modely v nich

vytvořené, například volně dostupný program Blender, či komerčně velmi rozšířený program Autodesk 3DS Max a podobné další programy. Součástí OBJ souboru bývají i MTL soubory, definující vlastnosti materiálu dané geometrie. Ve frameworku však zatím není toto rozšíření implementováno. V příloze A.1 je příklad struktury souboru typu OBJ, konkrétně se jedná o reprezentaci geometrie krychle. Řádky uvozené písmenem *v* reprezentují pozici jednotlivých vrcholů geometrie (vertex) ve 3D prostoru (x, y a z koordináty v daném pořadí), řádky uvozené písmeny *vt* reprezentují UV koordináty texturových dat pro daný vrchol a řádky uvozené písmeny *vn* reprezentují koordináty normál pro daný vrchol. Podstatnou částí jsou řádky uvozené písmenem *f*, které udávají výčet stěn geometrie a to pomocí indexu do polí ostatních atributů (pozic, UV koordinát, normál). Stěna geometrie může být reprezentována jedním ze 4 způsobů - v závislosti na přítomnosti daných atributů - následovně:

- *f 1 2 3* - reprezentace stěny geometrie bez přítomnosti informace o normále a UV koordinátě (stěna je tedy tvořena 1., 2. a 3. bodem ve výčtu pozic vrcholů),
- *f 1/3 2/2 3/4* - reprezentace stěny bez přítomnosti informace o normále, přičemž informace o UV koordinátě je přítomna (stěna je tedy tvořena 1., 2. a 3. bodem ve výčtu pozic vrcholů a přidružené UV koordináty k danému bodu jsou na pozicích 3, 2 a 4 v daném výčtu UV koordinát),
- *f 1//3 2//2 3//4* - reprezentace stěny bez přítomnosti informace o UV koordinátě příslušné k danému bodu, ale s přítomností informace o normále,
- *f 1/2/3 2/3/2 3/2/4* - reprezentace stěny s údaji o pozici, UV koordinátě i normále příslušící danému bodu.

Pro potřeby frameworku byl tedy vytvořen parser souborového typu OBJ. Parser je reprezentován třídou *ObjParser*, jež je interně používána ve třídě *ObjectReader*, jehož statická metoda *ReadObject(InputStream inputObjFileStream)* vrací objekt třídy *Model*. Třída *Model* ve frameworku poskytuje abstrakci nad libovolným trojrozměrným modelem, reprezentovaným údaji o pozicích jeho vrcholových bodů, UV koordinát a normál přidružených k danému bodu. Na následující ukázce lze vidět příklad použití statické metody *ReadObject(InputStream inputObjFileStream)* pro načtení informace z externího souboru *untitled.obj* přidruženého k sestavení aplikace:

```
public void onSurfaceChanged(GL10 gl, int width, int height)
{
    Model model = null;
```

```
    try
    {
        model = ObjectReader.ReadObject(context.getAssets().open("obj/untitled.obj"));
    } catch (IOException e)
    {
         ...
    }
     ...
}
```

Výpis 7: Ukázka použití statické metody *ReadObject* třídy *ObjectReader*

5.6.7 Ozvučení aplikace

Ozvučení aplikace je poměrně důležitým prvkem u aplikací zaměřených na rozvoj kognitivních schopností. Pro ozvučení aplikace existují v operačním systému 3 různé metody, každá vhodná pro určitý způsob ozvučení. Tyto metody jsou v API operačního systému reprezentovány třídami MediaPlayer, SoundPool a AudioTrack.

5.6.7.1 MediaPlayer Třída MediaPlayer je poměrně vysokoúrovňová, poskytující možnost přehrání jak audio, tak i video souborů. Umožňuje také přehrávání nejen z fyzického média (audio či video soubor uložený přímo v zařízení), ale i streamování média přes HTTP protokol. Typickým použitím třídy MediaPlayer jsou dlouho hrající zvukové či video soubory, jelikož třída nabízí rozhraní i například pro pozicování stopy a interně je zajištěno načítání delší stopy po blocích do vyrovnávací paměti. V aplikaci by třída mohla být použita pro přehrání hudby v pozadí. Pro ozvučení uživatelského rozhraní, čímž je ve vzorové aplikaci myšleno například ozvučení dotykového gesta, úspěchu, neúspěchu, či indikace tahu počítače a uživatele, je však třída MediaPlayer, kvůli své poměrně velké náročnosti na paměť a kvůli mírnému zpoždění, které se může projevit po spuštění přehrávání, zcela nevhodnou.

```
MediaPlayer backgroundMusicPlayer;
backgroundMusicPlayer = MediaPlayer.create(this, R.raw.music1);
backgroundMusicPlayer.setLooping(true);
backgroundMusicPlayer.start();
```

Výpis 8: Ukázka použití třídy MediaPlayer

5.6.7.2 SoundPool Třída SoundPool je vhodná pro přehrávání krátkých zvukových efektů. Třída SoundPool na rozdíl od třídy MediaPlayer neumožňuje přehrávání video souborů. Je vhodná pro přehrávání předem známého počtu zvukových souborů, typickým využitím ve vzorové aplikaci jsou tedy ozvučení uživatelského rozhraní (indikace kliknutí uživatele, indikace tahu, úspěchu či neúspěchu a podobně). Kromě základních operací načtení a přehrání zvuku poskytuje SoundPool následující operace:

- nastavení maximálního počtu zvuku, které lze simultánně přehrát
- pozastavení/zastavení zvuku
- zvukové smyčky
- změna rychlosti přehrávaní zvuku
- nastavení hlasitosti

Jelikož metoda *load* třídy SoundPool vrací po nahrání zvuku pouze interně generované *ID* ve formě proměnné typu *int*, byla v pomocném frameworku implementována pro snazší použití třída SoundHandler, zastřešující referenci na daný nahraný zvuk objektu třídy SoundPool.

```
// pouziti tridy SoundManager, obalujici konstrukci
// objektu tridy SoundPool
SoundManager manager = new SoundManager(MAX_NR_SOUNDS);

// pouziti tridy SoundHandler pro uchovani reference
// na zvukovy soubor nahrany tridou SoundManager
SoundManager.SoundHandler soundHandle;
soundHandle = manager.loadSound(context, R.raw.custom_sound);

// prehrani zvuku
manager.playSound(applauseHandle);
```

Výpis 9: Ukázka použití třídy SoundPool v pomocném frameworku

5.6.7.3 AudioTrack Třída AudioTrack poskytuje nízkoúrovňový přístup k přehrávání zvukových stop. Na rozdíl od tříd SoundPool a MediaPlayer poskytuje API pro přehrávání surových dat v podobě například pole bytů. Pro potřeby frameworku nebyla třída použita.

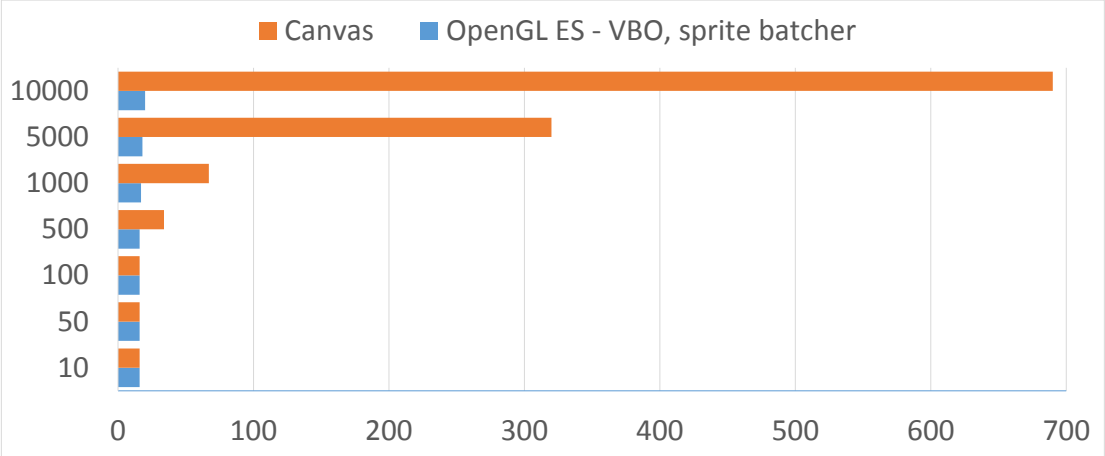
5.7 Výkonnostní testování frameworku

V rámci diplomové práce proběhlo testování vykreslovacích možností implementovaného frameworku. Referenčním vzorem byla u testů výkonnosti implementace založená na třídě *Canvas* operačního systému Android, sloužící pro zobrazování grafických prvků. Testovacím scénářem byla úloha vykreslení grafických primitiv typu *Sprite* (v případě OpenGL ES implementace se jedná konkrétně o 4 vrcholová data pro jednu instanci třídy Sprite, u implementace pro třídu Canvas je vnitřní implementace odstíněna od uživatele). Testování proběhlo ve vzorových aplikacích a to na vzorku s počtem instancí nabývající hodnoty 10, 50, 100, 500, 1000, 5000 a 10000. Pozorovanou veličinou byla doba překreslení jednoho snímku v kontinuálně překreslovaném režimu (jak již bylo zmíněno, doba překreslování je u operačního systému Android omezena na 60 snímků za sekundu, čemuž odpovídá přibližně doba 16 milisekund na vykreslení jednoho snímku). Testováno bylo jednak vykreslování statické geometrie (tedy geometrie, jejíž poloha v prostoru se nijak nemění) a také dynamické geometrie (tedy v každém vykreslovaném snímku byla změněna poloha geometrie v prostoru). Testování proběhlo na zařízení Asus EEE Pad Transformer TF101, jenž má následující parametry:

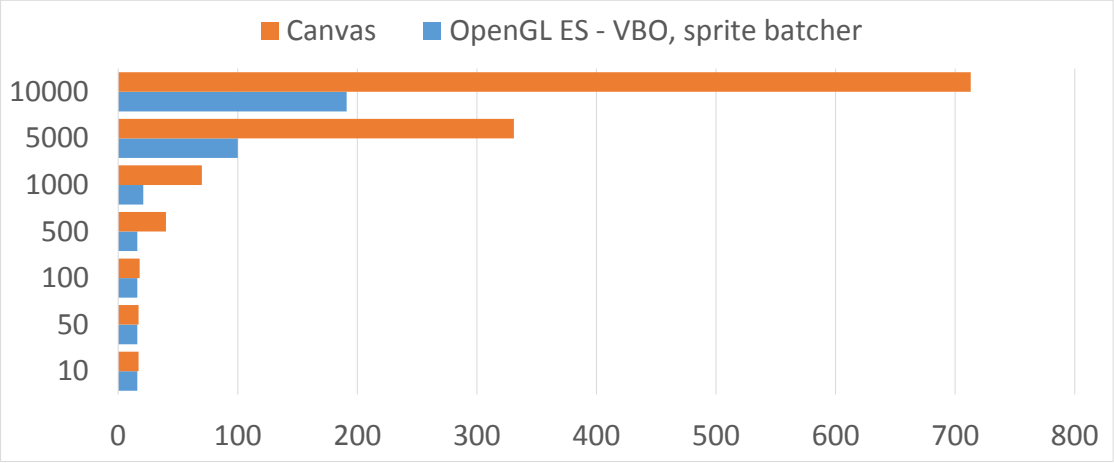
- Operační systém - Android verze 4.0.3
- Displej - úhlopříčka 25.6 cm, rozlišení 1280x800 obrazových bodů
- CPU - NVIDIA Tegra, dvou jádrový, frekvence 1.00 GHz
- Operační paměť - 1GB DDR2
- GPU - ULP GeForce

Na obrázcích 13 a 14 lze v podobě grafu vidět výsledky testování vykreslování statické respektive dynamické geometrie. Svislá osa představuje počet vykreslených entit v jednom snímku a vodorovná osa představuje čas vykreslení jednoho snímku v jednotkách milisekund. Na výsledcích si lze povšimnout poměrně značných rozdílů ve vykreslování statické geometrie. Příčinami je jednak optimalizace frameworku pro vykreslování velkého množství statické geometrie pomocí již zmíněné metody sprite batchingu a také použitím VBO, tedy konstruktu standardu OpenGL ES, jenž umožňuje geometrii přesunout na videopaměť zařízení a odpadá tedy přesun dat při každém vykresleném snímku. U testování jak statické, tak dynamické geometrie se ukázala špatná škálovatelnost použití třídy Canvas, kdy při vysokém počtu vykreslených entit dochází k razantnímu nárůstu času vykreslení snímku. Ukázalo se tedy, že zvolená implementace je vysoce optimalizovaná pro statické i dynamické vykreslování geometrie v porovnání s použitím třídy

Canvas. Tato výhoda se však projeví až při větším počtu vykreslené geometrie - v řádu stovek.



Obrázek 13: Výkonnostní testování - statická geometrie



Obrázek 14: Výkonnostní testování - dynamická geometrie

6 Uživatelské rozhraní aplikace

V rámci diplomové práce byl kladen důraz i na použitelnost uživatelského rozhraní. Pojem, jenž nám klasifikuje, zda vůbec, a do jaké míry je uživatelské rozhraní vhodné pro dané uživatele aplikace. Jak již bylo zmíněno v kapitole 4, uživatelské rozhraní bylo koncipováno velmi jednoduše, po vzoru některých dostupných aplikací, aby mělo co možná nejmenší rušivý vliv pro uživatele a aby středobodem aplikace byly pouze dané úkoly.

6.1 Použitelnost uživatelského rozhraní aplikace

Jelikož je aplikace primárně určena pro rozvoj kognitivních schopností uživatelů s určitou formou mentální dysfunkce, uživatelské rozhraní by mělo splňovat určité kvality odpovídající aplikacím podobného zaměření, aby nestálo v cestě samotnému užívání aplikace.

Důraz na použitelnost uživatelského rozhraní aplikace by měl být kladen zejména z důvodů nediskriminace určité skupiny uživatelů před efektivním použitím aplikace, před použitím aplikace k jejímu původnímu účelu, nebo aby použití aplikace nevedlo k nepohodlí či frustraci uživatele, což by pochopitelně mohlo vést k nelibosti uživatele aplikaci používat a v důsledku toho ke snížení schopnosti aplikace rozvinout dané kognitivní schopnosti [17].

Pro samotné měření efektivity prováděných úkonů, efektivnosti vykonávání úkonů v aplikaci a obecné spokojenosti uživatele s aplikací, máme k dispozici metriky použitelnosti, které nám pomohou ohodnotit použitelnost aplikace nikoliv pouze na základě domněnek tvůrce aplikace, nýbrž na základě prokazatelných dat a statistického měření. Požadavkem těchto metrik jsou poté následující 2 hodnoty:

- Pozorovatelnost (například schopnost určit zda daný úkol byl či nebyl splněn)
- Kvantifikovatelnost (možnost měřit tuto metriku absolutně)

Příklad pozorovatelné a kvantifikovatelné události metriky: ”90% uživatelů je schopno dokončit daný soubor úkonů v čase pod jednu minutu”.

Pro účely testování bylo na základě definice použitelnosti aplikace dle Jakoba Nielsena [17] stanoveno několik kvalitativních atributů (komponent) míry použitelnosti aplikace a to konkrétně:

- Naučitelnost: Tato komponenta nám umožňuje reflektovat to, jak je pro uživatele jednoduché respektive složité vykonat jednoduché předem definované úkoly při prvním setkání uživatele s tímto úkolem,

- Výkonnost: Komponenta reflektující rychlost úspěšného zvládnutí úkolu poté, co se již seznámil s jejími prvky,
- Zapamatovatelnost: Komponenta reflektující schopnost uživatele vykonat znovu úspěšně úkol poté, co se k úkolu po určité době neprovádění tohoto úkolu vrátil,
- Chybovost: Komponenta reflektující to, kolika chyb při provádění úkolu se uživatel dopustil a také jak jsou tyto chyby závažné, popřípadě jak je pro uživatele jednoduché/složité se z těchto chyb dostat k provádění úkolu,
- Uspokojení: Poměrně subjektivní komponenta reflektující uživatelovu obecnou míru uspokojení s daným úkolem.

6.2 Metodika testování použitelnosti uživatelského rozhraní

Pro zkoumání použitelnosti aplikace existuje několik metod, přičemž tou nejzákladnější a v našem případě nejužitečnější je tzv. uživatelské testování (user testing). Principem tohoto testování jsou následující tři body:

- získání určitého reprezentativního vzorku uživatelů,
- provedení reprezentativních úkolů těmito uživateli,
- pozorování toho, co uživatelé dělají, kde mají v aplikaci obtíže s uživatelským rozhraním a na základě těchto poznatků následná úprava uživatelského rozhraní.

6.3 Testování aplikace v rámci spolupráce s komunitou iSEN

Ve spolupráci s komunitou iSEN proběhlo v průběhu jednoho měsíce testování uživatelského rozhraní, na jehož základě došlo k vyhodnocení toho, zdali vůbec, a do jaké míry je naše vzorová aplikace použitelná pro osoby s určitým typem mentálního deficitu. Cílovou skupinou poskytnutou komunitou iSEN pro testování byla skupina pěti dětí ve věku od 7 do 13 let, kteří trpěli převážně lehkým mentálním postižením. Po dobu jednoho měsíce probíhalo testování pod dozorem speciálně školených pedagogů, kteří dohlíželi na to, jak si cílová skupina vedla při užívání aplikace, a na základě předem definovaných testovacích scénářů poté ohodnocovala dané složky použitelnosti uživatelského rozhraní. V tabulkách 4 a 5 lze vidět souhrn testování komunitou iSEN pro jednotlivé úkoly vzorové aplikace.

Subjekt	Diamanty	Krychle	Rotování	Obrazce
Oblast: *Naučitelnost - Jak bylo pro testovaný subjekt obtížné první použití aplikace ?* *1 - Snadné* *2 - Snadné s obtížemi* *3 - Zvladatelné* *4 - Obtížně zvladatelné* *5 - Velmi obtížné*				
s1	3	4	2	4
s2	3	4	2	4
s3	3	4	3	5
s4	3	4	1	4
s5	3	5	2	4
Oblast: *Výkonnost - Po seznámení s aplikací a po prvním použití, jak byste ohodnotili obtížnost dosažení cíle?* *1 - Efektivní* *2 - Efektivní s obtížemi* *3 - Efektivní s většími obtížemi* *4 - Málo efektivní* *5 - Neefektivní*				
s1	3	5	3	4
s2	4	4	1	5
s3	4	4	4	3
s4	3	4	1	4
s5	2	4	2	5
Oblast: *Zapamatovatelnost - Pokud se dítě vrací k aplikaci po delší přestávce (např. druhý den), uveďte s ohledem na délku přestávky schopnost opět aplikaci používat.* *1 - Celkově rychlé vybavení* *2 - Rychlé vybavení* *3 - Středně rychlé* *4 - Pomalé* *5 - Velmi pomalé*				
s1	3	3	2	3
s2	3	3	1	4
s3	3	2	3	5
s4	3	2	1	4
s5	3	3	1	5

Tabulka 4: Shrnutí testování komunitou iSEN

Subjekt	**Diamanty**	**Krychle**	**Rotování**	**Obrazce**
Oblast: *Chybovost - Jak často se dítě dopustí chyby při plnění úkolu aplikace?* *1 - Vůbec ne* *2 - Občas* *3 - Středně* *4 - Často* *5 - Velmi často*				
s1	4	5	2	2
s2	5	5	2	4
s3	5	4	3	5
s4	4	5	1	4
s5	3	4	2	4
Oblast: *Spokojenost - Jak byste ohodnotili spokojenost dítěte s aplikací?* *1 - Velmi spokojené* *2 - Spokojené* *3 - Neutrální* *4 - Spíše spokojené* *5 - Nespokojené*				
s1	3	5	2	3
s2	4	4	4	4
s3	4	4	3	4
s4	3	4	3	5
s5	3	5	2	4

Tabulka 5: Shrnutí testování komunitou iSEN

Jak již lze vidět z výsledné tabulky hodnocení, některé úkoly aplikace měly vzhledem k cílové skupině poměrně značné nedostatky. Toto však nebylo vyloženě způsobeno špatným návrhem uživatelského rozhraní, ale spíše skutečností, že cílovou skupinou testování byla zvolena skupina, jejíž mentální věk nedosahoval kvůli určitým mentálním poruchám požadované výše. I přes biologický věk, který byl u testovaných subjektů okolo 10 let, byl mentální věk testovaných subjektů mnohem nižší, a tudíž některé úlohy byly pro ně zcela nezvladatelné. Největší problém dělaly testovaným subjektům aplikace Obrazce a Krychle. Koncepčně byly tyto úlohy pro danou skupinu velmi těžce zvladatelné a sub-

jekty mnohdy ani nebyly schopny pochopit princip úlohy. Jedinou poměrně zvladatelnou úlohou byla úloha Rotování. Přínos této úlohy vidí speciální pedagogové komunity iSEN zejména v možném rozvoji prostorové orientace, soustředění a zrakové diferenciaci. Velkou výhodou je poté jednoduchost uživatelského rozhraní. Potvrdilo se tedy to, že dbát na jednoduchost je u aplikací podobného zaměření podstatné, aby samotné uživatelské rozhraní neodrazovalo uživatele od úkolu.

Kromě samotného hodnocení jednotlivých kategorií měli školení pedagogové prostřednictvím dotazníku možnost vyjádřit se k daným úkolům a například napsat věcné připomínky k hodnocení, v čem vidí přínos aplikace a případně různé náměty na zlepšení. Ukázalo se, že kupříkladu u úkolů Krychle a Diamanty by se pouhým doplněním tónů různé výšky (například při otočení krychle respektive při vysvícení diamantu) dala aplikace zacílit i na audiální rozvoj. Většina pedagogů vyjádřila neschopnost dítěte provést úkol Obrazce při zátěžové hodnotě n větší než 1. Pro již zmíněný mentální věk testovaných subjektů je toto zcela pochopitelné. Nicméně i při zátěžové hodnotě n rovné 1 byla úkolu vytknuta přílišná rychlost měnění obrazců Naopak jako pozitivum pedagogové shledali srozumitelnost obrazců a opět jednoduchost uživatelského rozhraní.

6.4 Změny v uživatelském rozhraní na základě výsledků testování

Samotné testování poskytlo velmi cenné poznatky, na základě kterých mohou být provedeny změny v uživatelském rozhraní, případně v samotném principu jednotlivých úkolů. Je však třeba brát v úvahu skutečnost, že pro cílovou skupinu by byly vhodné odlišné typy úkolů. Pozitivní je, že pro všechny navrhované změny jsou ve frameworku aplikace přichystány potřebné konstrukty, a tedy případné změny je poměrně jednoduché dodělat. Také se uzázalo, že i přes nevhodnou cílovou skupinu se některé aplikace ukázaly jako poměrně použitelné.

7 Testování aplikace na různých zařízeních

Jelikož při tvorbě aplikace pro operační systém Android v podstatě zacilujeme aplikaci na velké množství různých druhů přístrojů vyskytujících se na dnešních trzích, měla by aplikace splňovat určité prvky přenositelnosti. Přenositelnost prvků uživatelského rozhraní operačního systému Android nám nativně zajišťuje samotný systém, podporu různých velikostí displeje však standard OpenGL ES jako takový nezajišťuje, je tedy třeba dbát na určité zásady při vytváření grafických entit.

7.1 Podpora různých typů zařízení

Pro aplikaci bychom mohli stanovit jakési základní požadavky na přístroj, jako například minimální velikost úhlopříčky displeje, minimální požadovaná verze standardu OpenGL ES (v našem případě aplikaci nelze spustit na přístroji bez oficiální podpory OpenGL ES minimálně verze 2.0). Tyto parametry se stanovují v aplikačním manifest souboru, viz 5.2.2.

7.1.1 Density independent pixel

Pro potřeby pozicování a škálování prvků uživatelského rozhraní je v operačním systému Android interně používána jednotka Density independent pixel. Density independent pixel - dále jen dp - virtuální jednotka, která se v operačním systému Android používá na definování rozměrů a pozicí u prvků uživatelského rozhraní poskytnutých samotným operačním systémem. Jeden *density independent pixel* je ekvivalentem jednomu skutečnému obrazovému bodu u obrazovky s hustotou obrazových bodů 160 dpi. Operační systém Android se tedy interně stará o škálování zobrazených prvků podle jednoduchého přepočtu $px = dp * (dpi/160)$. Pro definování rozměrů a pozic prvků uživatelského rozhraní by se tato jednotka měla používat prioritně, i když k dispozici je i definování pomocí jednotek pixelů, v takovém případě se však výsledné uživatelské rozhraní stane špatně přenositelným.

7.1.2 Nezávislost aplikace na rozlišení a velikosti obrazovky zařízení

Prvky uživatelského rozhraní operačního systému Android, jako například tlačítka menu, různé seznamy a podobně, jsou tedy interně pozicovány a škálovány samotným operačním systémem, uživatel pouze definuje jejich pozici, velikost a různé další atributy v jednotkách *dp*. Knihovna OpenGL ES pochopitelně provázána s operačním systémem v

tomto smyslu není, definování různých grafických entit je tedy plně v kompetenci programátora. V aplikaci jsou veškeré velikosti a pozice grafických entit zobrazené pomocí OpenGL ES API (jako například již zmíněné texty, sprity) vypočítávány z rozlišení displeje zařízení. Tímto je zachována konzistentní velikost napříč zařízeními všech možných rozměrů. Nejsou tedy používány absolutní hodnoty, ale spíše procentuální podíly šířky a výšky obrazovky, tedy hodnoty z rozlišení displeje odvozené.

Pro ověření přenositelnosti byla aplikace testována na několika různých zařízeních. Konkrétně se jednalo o zařízení telefonního přístroje Sony Xperia Tipo Dual, jenž poskytuje velikost obrazovky 3.2 palce při rozlišení 320x480 obrazových bodů, dále na zařízení Asus EEE Pad Transformer, poskytující obrazovku o velikosti 10,1 palců při rozlišení 1280x800 obrazových bodů a dále poté několik různých zařízení prostřednictvím emulace na stolním PC. Pochopitelně je třeba brát v úvahu, že aplikace tohoto typu není vhodná na příliš malá zařízení, pro dobrou použitelnost je vhodná pro zařízení nabízející obrazovku minimálně o velikosti 10 palců. Testování na mobilním telefonu Sony Xperia proto proběhlo pouze pro prokázání přenositelnosti. Na žádném z testovaných přístrojů se neobjevily problémy s přenositelností uživatelského rozhraní.

8 Závěr

Cílem této práce bylo provést rešerši dostupných aplikací pro rozvoj kognitivních a motorických funkcí jedince a na základě získaných poznatků se pokusit implementovat vzorovou aplikaci pomocí grafického standardu OpenGL ES pro mobilní zařízení operačního systému Android. Na základě vybraných metod pro rozvoj kognitivních schopností byly v aplikaci implementovány čtyři různé úkoly.

Pro potřeby snadnější implementace aplikace byl vytvořen framework poskytující potřebné konstrukty, které zjednodušují práci s grafickými primitivy standardu OpenGL a zastřešující různé další složky tvorby interaktivní aplikace, jako například práce se zvukovými soubory, zobrazování textu, vstupy/výstupy a podobně. Cílem tvorby frameworku nebylo konkurovat současným volně dostupným a komerčním frameworkům, ale spíše šlo o vytvoření jednoduchého frameworku poskytujícího možnost rychlejšího vývoje aplikací s grafickým uživatelským rozhraním s 2D i 3D prvky. Framework byl vyvinut z důvodu toho, že většina komerčních frameworků je buďto na podobný typ aplikací zbytečně komplexních, případně jejich licence neumožňuje volné použití a naopak volně dostupné frameworky neposkytují jednoduše možnost zobrazení 3D prvků, ale zaměřeny jsou pouze na jednoduché 2D aplikace, navíc svou koncepcí jsou mnohdy zacíleny na určité konkrétní typy her. Úkoly ukázkové aplikace byly implementovány právě pomocí pomocného frameworku. Výkonnostním testováním frameworku se ukázalo, že je postačující i na zobrazování velkého množství geometrie.

Samotný framework, vytvořený v rámci diplomové práce, je prozatím velmi jednoduchý a je zde spousta prostoru pro jeho rozšíření v případě dalšího použití. Možnými dalšími rozšířeními jsou například implementace fyzikálního systému, buďto s využitím některé z knihoven jako například Box2D či Bullet, případně plnou vlastní implementací. Případným dalším rozšířením by mohlo být připravení aplikace na různé druhy alternativních augmentativních vstupů.

Použitelnost vzorové aplikace byla testována v rámci spolupráce s komunitou iSEN prostřednictvím uživatelského testování na konkrétní cílové skupině. Ukázalo se však, že dodaná cílová skupina nebyla pro testování vhodná. Příliš nízký mentální věk testovaných subjektů se poté negativně odrazil v hodnocení použitelnosti aplikace. Bohužel již nebyla možnost testovat aplikaci na jiné cílové skupině. I přesto však z testování vzešly mnohé poznatky o uživatelském rozhraní aplikace pro uživatele s velmi nízkým mentálním věkem, na které je třeba při tvorbě podobných aplikací brát ohled. Ukázalo se také, že i přes nízký mentální věk testovaných subjektů byly některé úkoly cílovou skupinou použitelné a mohly by tedy sloužit jako základ pro budoucí vývoj.

9 Reference

[1] CHEN, S. H. A., J. D. THOMAS, R. L, Nancy CARNEY a Hugo du COUDRAY. *The effectiveness of computer-assisted cognitive rehabilitation for persons with traumatic brain injury.* Brain Injury. 1997, vol. 11, issue 3, s. 295-318. DOI: 10.1176/appi.books.9781585624201.687745.

[2] CICERONE, Keith D., Cynthia DAHLBERG, Kathleen KALMAR, Donna M. LANGENBAHN, James F. MALEC, Thomas F. BERGQUIST, Thomas FELICETTI, Joseph T. GIACINO, J.Preston HARLEY, Douglas E. HARRINGTON, Jean HERZOG, Sally KNEIPP, Linda LAATSCH a Philip A. MORSE. *Evidence-based cognitive rehabilitation: recommendations for clinical practice.* Archives of Physical Medicine and Rehabilitation. 2000, vol. 81, issue 12, s. 1596-1615. DOI: 10.1053/apmr.2000.19240.

[3] FALCONER J. *Computers and brain injury: some guidelines for rehabilitation.* Computers and brain injury: some guidelines for rehabilitation. [online]. [citováno 2015-01-11]. URL: www.brain-train.com/articles/computer/htm

[4] LOPRESTI, Edmund Frank, Alex MIHAILIDIS a Ned KIRSCH. *Assistive technology for cognitive rehabilitation: state of the art.* Neuropsychological Rehabilitation. 2004, vol. 14, 1-2, s. 5-39. DOI: 10.1080/09602010343000101.

[5] LYNCH, Bill. *Historical review of computer-assisted cognitive retraining.* Journal of Head Trauma Rehabilitation. 2002, vol. 17, issue 5.

[6] MATTHEWS, Charles G., J. Preston HARLEY a James F. MALEC. *Guidelines for computer-assisted neuropsychological rehabilitation and cognitive remediation.* Clinical Neuropsychologist. 1991, vol. 5, issue 1, s. 3-19. DOI: 10.1080/13854049108401837.

[7] PARK, Norman W. a Janet L. INGLES. *Effectiveness of attention rehabilitation after an acquired brain injury: a meta-analysis.* Neuropsychology. 2001, vol. 15, issue 2, s. 199-210. DOI: 10.1037//0894-4105.15.2.199.

[8] SOHLBERG, McKay Moore a Catherine A MATEER. *Introduction to cognitive rehabilitation: theory and practice.* New York: Guilford Press, c1989, xviii, 414 p. ISBN 08-986-2738-9.

[9] HUCKANS, Marilyn, Lee HUTSON, Elizabeth TWAMLEY, Amy JAK, Jeffrey KAYE a Daniel STORZBACH. *Efficacy of Cognitive Rehabilitation Therapies for Mild Cognitive*

Impairment (MCI) in Older Adults: Working Toward a Theoretical Model and Evidence-Based Interventions. Neuropsychology Review. 2013, vol. 23, issue 1, s. 63-80. DOI: 10.1007/s11065-013-9230-9.

[10] HUANG-POLLOCK, CYNTHIA L., a Sarah L. KARALUNAS. *Working Memory Demands Impair Skill Acquisition in Children with ADHD.* Journal of Abnormal Psychology 119.1 (2010): 174-85.

[11] KLINGBERG, TORKEL, Elisabeth FERNELL, Pernille J. OLESEN, Mats JOHNSON, Per GUSTAFFSON, Kerstin Dahlström, Christopher G. GILLBERG, Hans FORSSBERG, and Helena WESTERBERG. *Computerized Training Of Working Memory In Children With ADHD-A Randomized, Controlled Trial.* Journal of the American Academy of Child & Adolescent Psychiatry: 177-86.

[12] JOHNSON, A. MICHAEL. Speed of mental rotation as a function of problem-solving strategies. Perceptual and Motor Skills. 71(3): 803-806. DOI: 10.2466/pms.1990.71.3.803.

[13] *Dashboards - Android Developers* [online]. [citováno 20.3.2014] URL: http://developer.android.com/about/dashboards/index.html.

[14] MEIER, Reto. *Professional Android 4 application development. [3rd ed.].* Indianapolis, IN.: Wiley/[Wrox], 2012, xlii, 817 p.

[15] LEVIN, Jonathan. *Android Internals::Power User's View.* Jonathan Levin; 2015. 1. edice. ISBN 0991055527.

[16] ZECHNER, Mario. *Beginning Android 4 games development.* Berkeley: Apress, c2011, xv, 677s. ISBN 978-1-4302-3987-1.

[17] *Usability 101: Introduction to Usability.* [online]. [citováno 7.3.2015]. URL: http://www.nngroup.com/articles/usability-101-introduction-to-usability/.

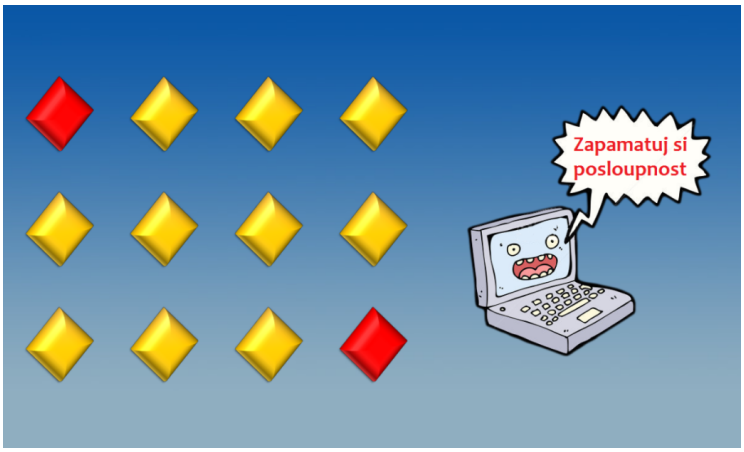
[18] HORŇÁK, Roman *Grafický engin pro tvorbu Android aplikací.* (Vysoká škola báňská - Technická univerzita Ostrava. Fakulta elektrotechniky a informatiky, Bakalářská práce, 2014)

A Přílohy

A.1 Příklad souborového typu OBJ - krychle

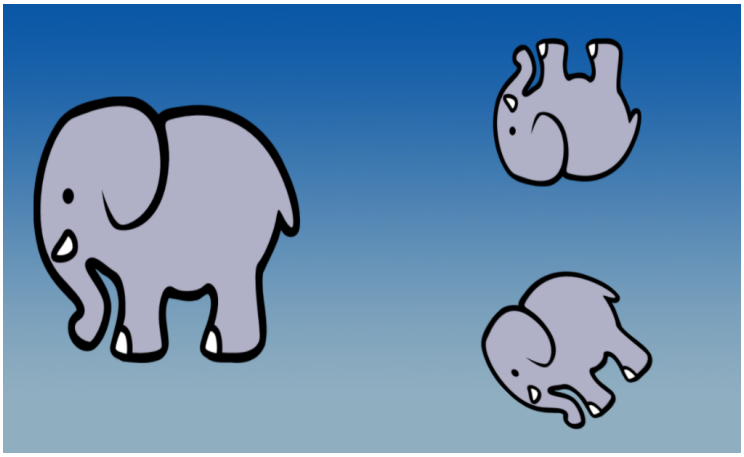
```
# Blender v2.69 (sub 0) OBJ File: 'untitled.blend'
# www.blender.org
o Cube
v 1.000000 -1.000000 -1.000000
v 1.000000 -1.000000 1.000000
v -1.000000 -1.000000 1.000000
v -1.000000 -1.000000 -1.000000
v 1.000000 1.000000 -0.999999
v 0.999999 1.000000 1.000001
v -1.000000 1.000000 1.000000
v -1.000000 1.000000 -1.000000
vt 0.499999 0.500000
vt 0.499999 0.749956
vt 0.250043 0.749957
vt 0.250042 0.500001
vt 0.499998 0.250043
vt 0.250041 0.250044
vt 0.250040 0.000088
vt 0.499996 0.000087
vt 0.749956 0.749956
vt 0.749956 0.999913
vt 0.500000 0.999913
vt 0.250043 0.999913
vt 0.000087 0.999913
vt 0.000087 0.749957
usemtl Cube
s off
f 1/1 2/2 3/3 4/4
f 5/5 8/6 7/7 6/8
f 1/9 5/10 6/11 2/2
f 2/2 6/11 7/12 3/3
f 3/3 7/12 8/13 4/14
f 5/5 1/1 4/4 8/6
```

A.2 Výsledné uživatelské rozhraní úkolů aplikací



Obrázek 15: Uživatelské rozhraní úkolu Diamanty ve výsledné aplikaci

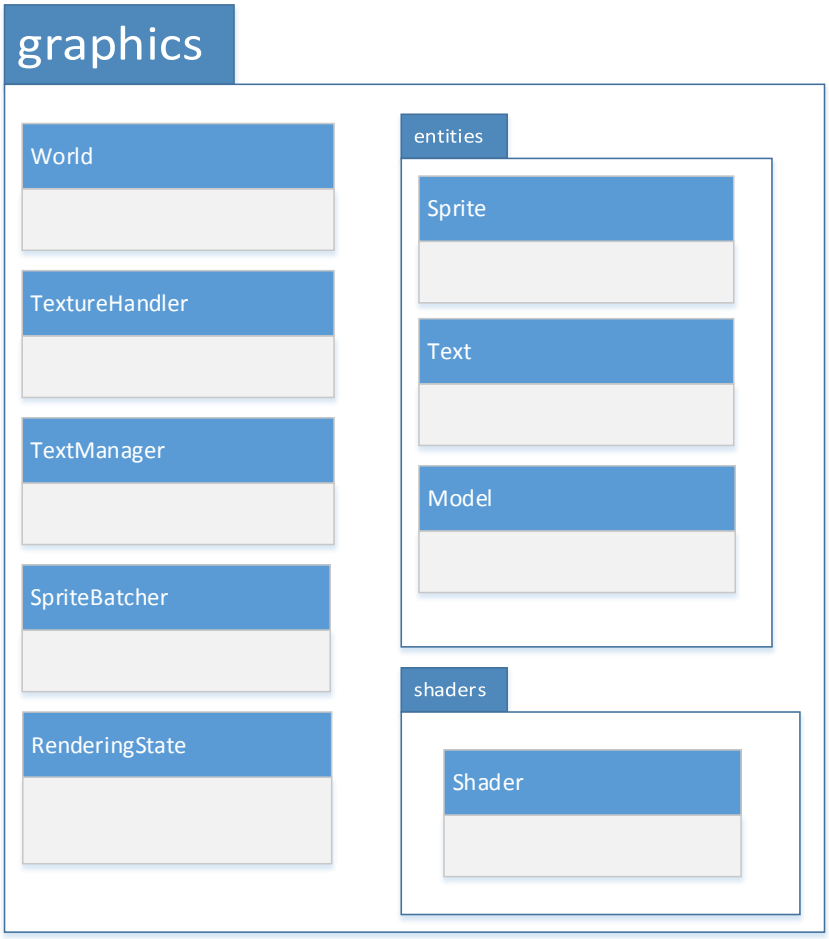
Obrázek 16: Uživatelské rozhraní úkolu Krychle ve výsledné aplikaci



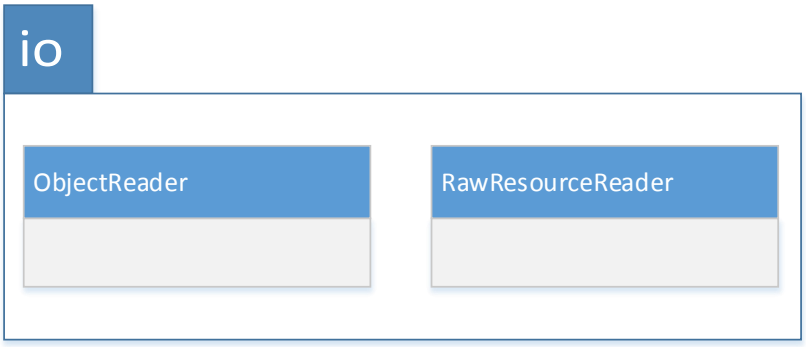
Obrázek 17: Uživatelské rozhraní úkolu Rotování ve výsledné aplikaci

Obrázek 18: Uživatelské rozhraní úkolu Obrazce ve výsledné aplikaci

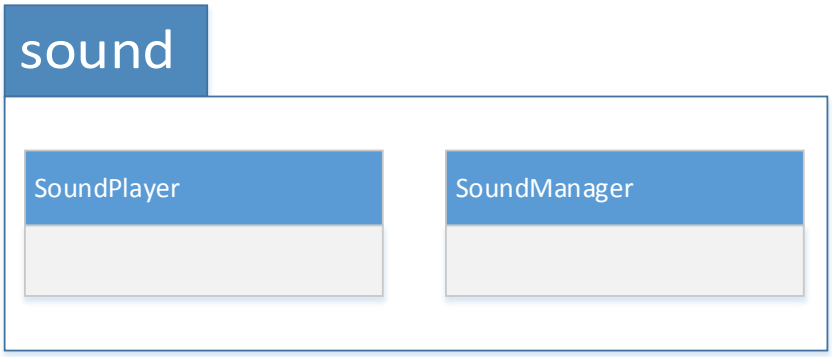
A.3 Diagramy balíčků jednotlivých částí frameworku



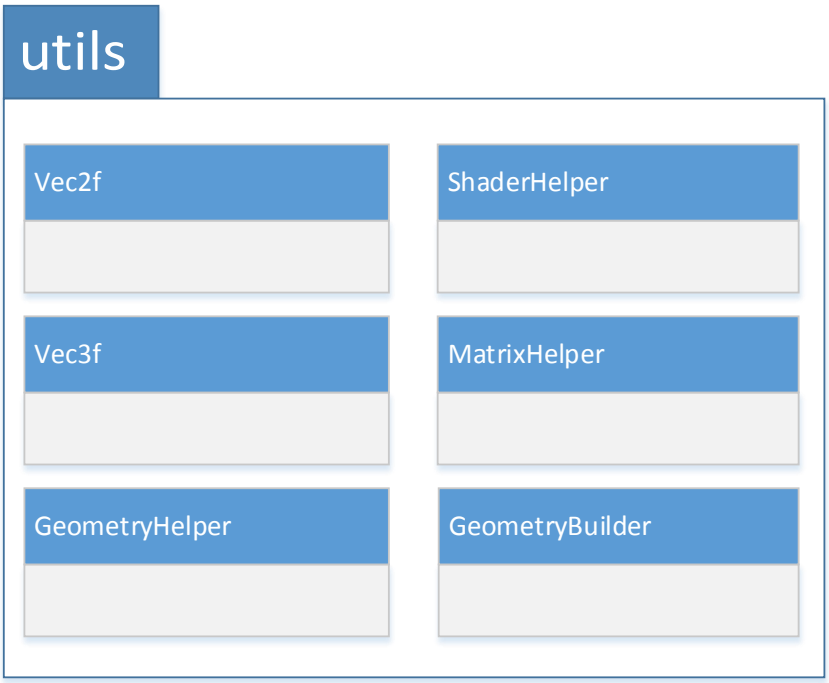
Obrázek 19: Diagram balíčků části frameworku *graphics*



Obrázek 20: Diagram balíčků části frameworku *io*



Obrázek 21: Diagram balíčků části frameworku *sound*



Obrázek 22: Diagram balíčků části frameworku *utils*

A.4 Obsah CD

Adresář	Popis
Src	Kompletní projektové řešení vzorové aplikace pro integrované vývojové prostředí Android Studio
Text	Text diplomové práce